МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Рубцовский индустриальный институт (филиал)
федерального государственного бюджетного образовательного
учреждения высшего образования
«Алтайский государственный технический университет им. И.И. Ползунова»
(РИИ АлтГТУ)

Т.М. ОБУХОВИЧ
Л.А. ПОПОВА

ПРОГРАММИРОВАНИЕ. ПАСКАЛЬ

Учебное пособие
для студентов направления
«Информатика и вычислительная техника»

Рекомендовано Рубцовским индустриальным институтом (филиалом) ФГБОУ ВО «Алтайский государственный технический университет им. И.И. Ползунова» в качестве учебного пособия для студентов, обучающихся по направлению подготовки «Информатика и вычислительная техника»

Рубцовск 2015

УДК 681.3.06

Обухович Т.М., Попова Л.А. Программирование. Паскаль: Учебное пособие для студентов направления «Информатика и вычислительная техника» / Рубцовский индустриальный институт. – Рубцовск, 2015.– 73 с.

Рассмотрены основные команды и операторы языка Паскаль. Приведены примеры решения задач по изложенному материалу. Прилагаются задания к практическим и лабораторным работам, а также вопросы для самостоятельной подготовки к промежуточной аттестации.

Пособие предназначено для проведения практических и лабораторных работ по курсу «Программирование» для студентов направления «Информатика и вычислительная техника» 1-го курса.

Рассмотрено и одобрено на заседании НМС Рубцовского индустриального института.
Протокол №8 от 26.11.15.

Рецензент: доцент кафедры
высшей математики РИИ Г.А. Обухова

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Предлагаемое учебное пособие «Программирование. Паскаль» написано в соответствии с программой курса «Программирование» для студентов направления «Информатика и вычислительная техника» и предназначено для аудиторной и самостоятельной работы студентов по курсу.

Учебное пособие содержит теоретический курс по основам программирования на языке Паскаль, примеры решения задач, задания к лабораторным работам и практическим занятиям, а также вопросы для подготовки к промежуточной аттестации.

1. ЯЗЫК ПРОГРАММИРОВАНИЯ

Программирование – это искусство получения ответов от машины.

При этом необходимо знать, какого рода вопросы мы имеем право задавать. Ответом может быть число или текст. Мы не можем попросить найти общее решение дифференциального уравнения, но можем получить частное решение, удовлетворяющее конкретным условиям.

При программировании нужно пройти три этапа:

1. Ясно и точно установить, что должно быть сделано.
2. Изобрести алгоритм, т.е. определенную последовательность действий, ведущую к желаемому результату.
3. Выразить алгоритм в таком виде, в котором его могла бы воспринять машина.

При обработке экономической информации шаги 1 и 2 называют системным анализом. Шаг 3 называют программированием или кодированием.

Чтобы создать программу, мало знать язык программирования. Программу надо сконструировать, разбить на определенные блоки и выстроить эти блоки один за другим в соответствии с заранее заданным порядком действий. Этот порядок и называется алгоритмом. У каждого алгоритма есть определенные характеристики (свойства):

1. *Понятность* – алгоритм должен включать только те команды, которые доступны исполнителю и входят в его систему команд.
2. *Массовость* – алгоритм составляется для целого класса задач и должен быть пригодным для разных наборов исходных данных.
3. *Результативность* (разрешимость) – при корректно заданных исходных данных алгоритм должен завершать работу и выдавать результат за конечное число шагов.
4. *Дискретность* – алгоритм должен представлять процесс решения задачи как последовательное выполнение некоторых простых шагов. При этом выполнение каждого шага алгоритма реализуется за конечный отрезок времени, то есть преобразование исходных данных в результат осуществляется *во времени дискретно*.

5. *Детерминированность* (определённость) – в каждый момент времени следующий шаг работы однозначно определяется значениями данных, полученными на предыдущих шагах. Таким образом, алгоритм выдаёт один и тот же результат для одних и тех же исходных данных.
6. *Направленность* – последовательность выполнения команд строго определена.

Таким образом, алгоритм дает механическую инструкцию для решения задачи. В этом причина того, что алгоритмы могут быть реализованы на ЭВМ.

Программа – это, по существу, задание алгоритма в форме, пригодной для автоматического выполнения. Выполнение может осуществляться двумя способами:

1. Можно проводить пооператорную (покомандную) обработку и выполнение исходной программы. Это называется *интерпретацией*.

2. Можно получить (преобразовать) программу на другом языке (машинный код). Этот процесс называется *компиляцией*.

Интерпретация медленнее и менее эффективна, чем компиляция.

Язык программирования – это средство связи между программистом и машиной. Язык обеспечивает концептуальную основу в двух отношениях: во-первых, имеется некоторый набор операций, во-вторых, имеются типы элементов и некоторая организация данных, которые подлежат обработке.

Алгоритмический язык содержит четыре основных элемента:

Символы языка – это основные неделимые знаки, в терминах которых пишутся все тексты на языке.

Элементарные конструкции – это минимальные единицы языка, имеющие самостоятельный смысл. Они образуются из основных символов языка.

Выражение в алгоритмическом языке состоит из элементарных конструкций и символов, оно задает правило вычисления некоторого значения.

Оператор задает полное описание некоторого действия, которое необходимо выполнить. Для описания сложного действия может потребоваться группа операторов. В этом случае операторы объединяются в *составной оператор* или *блок*.

Описание каждого элемента языка задается его *синтаксисом* и *семантикой*. Синтаксические определения устанавливают правила построения элементов языка. Семантика определяет смысл и правила использования тех элементов языка, для которых были даны синтаксические определения.

Действия, заданные операторами, выполняются над *данными*. Предложения алгоритмического языка, в которых даются сведения о типах данных, называются *описаниями* или неисполняемыми операторами.

Объединенная единым алгоритмом совокупность описаний и операторов образует *программу* на алгоритмическом языке.

2. УРОВНИ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Разные типы процессоров имеют разные наборы команд. Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности, то он называется *языком низкого уровня*. Операторы языка близки к машинному коду и ориентированы на конкретные команды процессора. К языкам низкого уровня относится язык Ассемблер. Этот язык программирования дает полный контроль над аппаратным обеспечением компьютера и генерирует очень эффективный исполняемый код. Однако программирование на этом языке требует больших затрат времени, он труден для изучения, и программы, написанные на Ассемблере, трудно отлаживать. Ассемблер сегодня используется в основном для написания системного ПО.

Языки программирования, имитирующие естественные языки, обладающие укрупненными командами, ориентированными на решение прикладных содержательных задач, называются *языками высокого уровня*.

Языки программирования высокого уровня имеют ряд достоинств:

- алфавит языка значительно шире машинного, что делает его гораздо более выразительным и существенно повышает наглядность и понятность текста;
- набор операций, допустимых для использования, не зависит от набора машинных операций, а выбирается из соображений удобства формулирования алгоритмов решения задач определенного класса;
- конструкции команд (операторов) задаются в удобном для человека виде;
- используется аппарат переменных и действий с ними;
- поддерживается широкий набор типов данных.

Таким образом, языки программирования высокого уровня являются машинно-независимыми и требуют использования соответствующих программ-переводчиков (трансляторов) для представления программы на языке машины, на которой она будет исполняться.

3. ПОКОЛЕНИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Для работы с компьютерами первого поколения программисты писали свои программы в машинных кодах (machine language) – то есть с помощью одних только нулей и единиц. Машинные коды были языком программирования первого поколения.

Второе поколение ознаменовалось появлением в начале 50-х годов языка программирования Ассемблера (assembly language). Вместо нулей и единиц теперь могли использоваться операторы, которые были похожи на слова английского языка. Компилятор преобразовывал эти выражения в машинные коды.

Третье поколение (относится к 60-м годам) отмечено появлением первых языков программирования высокого уровня (high-level languages). Эти языки впервые позволили решать задачи из разных областей. Для языков высокого уровня понадобились более быстрые, высокоэффективные компиляторы, поскольку при преобразовании исходного кода выходные программы получались большими.

Четвертое поколение языков программирования зародилось в конце 70-х, а развитие их продолжается по сей день. К ним относятся программные инструменты, которые позволяют пользователям разрабатывать программы, обладая минимальными техническими навыками. Языки постепенно становятся менее процедурными, они требуют от пользователя только указания, что надо сделать, без подробного описания, как это должно быть выполнено. Таким образом, с помощью непроцедурного языка можно выполнить некоторую задачу, описав меньшее число шагов, чем при решении той же задачи, пользуясь процедурным языком. Эти языки существенно уменьшили время разработки ПО.

4. ОБЗОР ЯЗЫКОВ ПРОГРАММИРОВАНИЯ ВЫСОКОГО УРОВНЯ

FORTRAN – первый компилируемый язык, созданный Джимом Бэкусом в 50-е годы, был спроектирован в основном для инженеров, математиков и ученых. На FORTRANe можно довольно просто описывать сложные вычисления, манипулировать массивами и распечатывать выходные множества чисел. Хотя на этом языке было написано немало бизнес-приложений, он не очень подходит для частых операций ввода-вывода и работы со списками. FORTRAN сравнительно легко осваивается, но его синтаксис очень требователен к точности ввода операторов, что вызывает частые ошибки и делает сложной отладку программ.

COBOL – компилируемый язык для применения в экономической области и решения бизнес-задач, разработанный в начале 60-х годов. Он отличается большой "многословностью"-его операторы выглядят как обычные английские фразы. В Коболе были реализованы очень мощные средства работы с большими объемами данных, хранящимися на различных внешних носителях. На этом языке создано очень много различных приложений, которые активно эксплуатируются и сегодня.

ALGOL – компилируемый язык, созданный в 1960 году. Он был призван заменить Фортран, но из-за более сложной структуры не получил широкого распространения. В 1968 году была создана версия Алгол68, по своим возможностям опережающая и сегодня многие языки программирования, однако из-за отсутствия достаточно эффективных компьютеров для нее не удалось своевременно создать хорошие компиляторы.

PASCAL – создан в конце 70-х годов Никлаусом Виртом, во многом напоминает Алгол, но в нем ужесточен ряд требований к структуре программы

и имеются возможности, позволяющие успешно применять его при создании крупных проектов.

BASIC – был создан в 1964 году, использовался в учебных целях, сегодня – один из самых используемых языков программирования. Для этого языка имеются и компиляторы, и интерпретаторы. Язык прост в изучении, его можно использовать даже для построения больших систем высокой производительности. Слабая сторона BASICa – то, что он выполняет все задачи одинаково, без оптимизации кода.

СИ – был создан в лаборатории Bell и первоначально не рассматривался как массовый. Он планировался для замены ассемблера, чтобы иметь возможность создавать столь же эффективные и компактные программы и в то же время не зависеть от конкретного вида процессора. Основная область его применения – коммерческие пакеты прикладных программ для микрокомпьютеров – серверов, рабочих станций и ПК.

СИ++ – это объектно-ориентированное расширение языка Си, созданное Бьярном Страуструпом в 80-х годах. Множество новых мощных возможностей, позволивших резко увеличить производительность программистов, наложилось на унаследованную от языка Си определенную низкоуровневость, в результате чего создание сложных и надежных программ потребовало от разработчиков высокого уровня профессиональной подготовки.

JAVA (Ява) – язык был создан на основе Си++. Он призван упростить разработку приложений на основе Си++, путем исключения из него всех низкоуровневых возможностей. Но главная особенность этого языка – компиляция не в машинный код, а в платформо-независимый байт-код. Этот байт-код может выполняться с помощью интерпретатора виртуальной машины Java-машины JVM(Java Virtyal Machine), версии которой созданы сегодня для любых платформ. Благодаря наличию Java-машин программы на Java можно переносить не только на уровне исходных текстов, но и на уровне обычного байт-кода, поэтому по популярности язык Ява сегодня занимает второе место в мире после Бейсика.

5. ЯЗЫК ПРОГРАММИРОВАНИЯ ПАСКАЛЬ

Алгоритмический язык высокого уровня Паскаль был разработан в конце 60-х годов швейцарским профессором Никлаусом Виртом (цюрихская высшая техническая школа).

Программирование на Паскаль стало весьма популярным благодаря тому, что в сочетании с высокоэффективным выходным кодом, генерируемым компилятором этого языка, программы, написанные на нем, занимают немного места в памяти.

К основным достоинствам языка следует отнести:

– гибкость;

– надежность;

– простота и ясность конструкций;

– легкость реализации на большинстве современных ЭВМ;

– возможность достаточно полного контроля правильности программы как на этапе компиляции, так и во время выполнения;

– возможность удовлетворения требованиям структурного программирования;

– возможность использования для программирования задач различных профилей;

– наличие набора структурных типов данных (массивов, записей, записей с вариантами, множеств, файлов);

– возможность построения новых типов данных.

6. ПОНЯТИЕ АЛГОРИТМА

Единого «истинного» определения понятия «алгоритм» нет.

Алгоритм – это последовательность действий, направленных на получение определённого результата за конечное число шагов.

Алгоритм – это точная, однозначная, конечная последовательность действий, которую должен выполнить пользователь для достижения конкретной цели либо для решения конкретной задачи или группы задач.

Алгоритм – это конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает пятью важными чертами: конечность, определённость, ввод, вывод, эффективность.

Алгоритм может быть записан словами и изображён схематически. Обычно сначала алгоритм описывается словами, но по мере приближения к реализации он обретает всё более формальные очертания и формулировку на языке, понятном исполнителю (например, машинный код). Для описания алгоритма применяются блок-схемы. Другим вариантом описания, независимым от языка программирования, является псевдокод.

Псевдокод – компактный (зачастую неформальный) язык описания алгоритмов, использующий ключевые слова языков программирования, но опускающий несущественные подробности и специфический синтаксис.

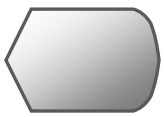
Псевдокод обычно опускает детали, несущественные для понимания алгоритма человеком. Такими несущественными деталями могут быть описания переменных, системно-зависимый код и подпрограммы. Главная цель использования псевдокода – обеспечить понимание алгоритма человеком, сделать описание более воспринимаемым, чем исходный код на языке программирования. Псевдокод широко используется в учебниках и научно-технических публикациях, а также на начальных стадиях разработки компьютерных программ.

Блок-схемы можно рассматривать как графическую альтернативу псевдокоду.

Правила выполнения схем определяются единой системой программной документации [3, 4, 5]. Данные документы, в частности, регулируют способы построения схем и внешний вид их элементов (табл.1).

Таблица 1

Основные элементы схем алгоритма

Наименование	**Обозначение**	**Функция**
Терминатор (пуск-останов)		Элемент отображает вход из внешней среды или выход из нее (наиболее частое применение − начало и конец программы). Внутри фигуры записывается соответствующее действие.
Процесс		Выполнение одной или нескольких операций, обработка данных любого вида (изменение значения данных, формы представления, расположения). Внутри фигуры записывают непосредственно сами операции. Например, операцию присваивания: a := 10*b + c.
Решение		Отображает решение с одним входом и двумя или более альтернативными выходами, из которых только один может быть выбран после вычисления условий, определенных внутри этого элемента. Например, условные операторы if и case.
Данные (ввод-вывод)		Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод).
Дисплей		Символ отображает данные, представленные в человекочитаемой форме на носителе в виде отображающего устройства (экран для визуального наблюдения, индикаторы ввода информации).
Документ		Символ отображает данные, представленные на носителе в удобочитаемой форме (ввод-вывод данных, носителем которых служит бумага).
Граница цикла		Символ состоит из двух частей − соответственно, начало и конец цикла − операции, выполняемые внутри цикла, размещаются между ними. Условия цикла и приращения записываются внутри символа начала или конца цикла − в зависимости от типа организации цикла.

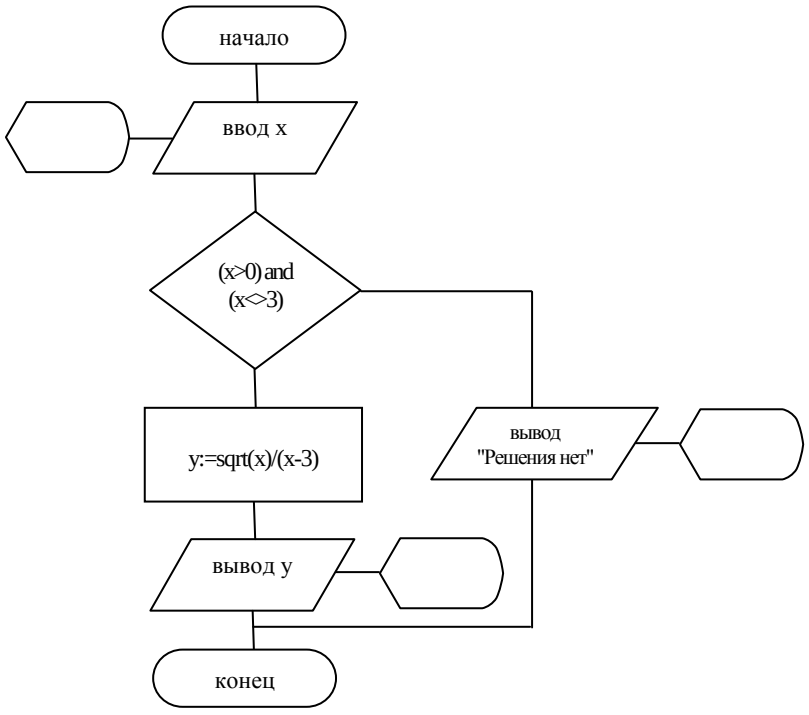
Продолжение таблицы 1

Основные элементы схем алгоритма

Наименование	**Обозначение**	**Функция**
Комментарий		Используется для более подробного описания шага, процесса или группы процессов. Описание помещается со стороны квадратной скобки и охватывается ей по всей высоте. Пунктирная линия идет к описываемому элементу либо группе элементов (при этом группа выделяется замкнутой пунктирной линией).

Описание других элементов схем можно найти в соответствующих ГОСТ [3, 4, 5].

Пример: Вычислить значение функции $y = \frac{\sqrt{x}}{x-3}$.

Блок-схема	Псевдокод
	начало ввод x если (x>0) and (x<>3) то y:= sqrt(x)/(x-3) вывод y иначе вывод "Решения нет" конец

7. АЛФАВИТ ЯЗЫКА. СТРУКТУРА ПРОГРАММЫ

Алфавит языка Паскаль включает:

1. буквы латинского алфавита ("a"–"z", "A"–"Z");
2. цифры (0–9);
3. специальные символы (+ – * _ . ; {} …) или пары символов (<> >= := …);

Программа на языке Паскаль состоит из заголовка и собственно программы, называемой блоком. Блок состоит из разделов. Максимальное число разделов – семь. Разделы, как правило, располагаются в следующем порядке.

1. Заголовок – **PROGRAM <имя> (файлы);**
2. Раздел подключаемых модулей – **USES <имя 1>, <имя 2>, …;**
3. Раздел меток – **LABEL <имя 1>, <имя 2>, …;**
4. Раздел констант – **CONST <идентификатор>=<значение>;**
5. Раздел описания типов данных **TYPE <имя типа>=<значение типа>;**
6. Раздел описания переменных **VAR <идентификатор 1, …>:<тип>;**
7. Раздел описания процедур и функций;
8. Раздел операторов **BEGIN…END.**

В любое место программы могут быть включены комментарии. Комментарии заключаются в скобки { } или (* *).

8. ВВОД И ВЫВОД ДАННЫХ

Ввод данных – это передача информации от внешнего носителя в оперативную память для обработки.

Вывод данных – обратный процесс.

Существуют стандартные файлы ввода и вывода: **INPUT** и **OUTPUT**, которые являются параметрами программы. Программа получает входные данные из файла **INPUT** и помещает результат обработки в файл **OUTPUT**. Файлу **INPUT** назначена клавиатура, **OUTPUT** – экран терминала.

Для выполнения операций ввода–вывода служат процедуры:

READ[LN](перем1, перем2, …); – обеспечивает ввод данных;

READLN; – приостановка работы программы до нажатия клавиши **Enter**.

Процедура **WRITE** производит вывод данных.

WRITE[LN](‘Строковая константа’, перем1, …); – обеспечивает вывод данных.

При выводе вещественного числа желательно указывать формат вывода. Например, если b=118.5, то при записи **WRITE(b:8:3);** получим результат – _118.500.

WRITELN; – вывод пустой строки или переход на новую строку.

9. КОНЦЕПЦИЯ ТИПОВ ДАННЫХ

Данные – это общее понятие для всего, чем оперирует ЭВМ, в которой они (данные) всегда представлены как последовательность двоичных разрядов. Введение абстрактных понятий, таких как вектор, запись, файл, множество и другие, в языки программирования высокого уровня поставило вопросы организации данных. Возникает вопрос: как различать отдельные объекты? Каждому объекту необходимо дать уникальное имя. А чтобы определить, что

данный объект принадлежит определенному классу, необходимо указать некоторое инвариантное свойство, присущее всем объектам данного класса. Тем самым мы пришли к понятию типа, который и является этим инвариантным свойством.

Тип в языках программирования – это множество значений плюс множество операций над этими значениями. Таким образом, определяя в программе некоторый тип, мы тем самым указываем:

1. какие значения могут принимать объекты данного типа;
2. какие операции над объектами разрешается выполнять.

В языке программирования Паскаль приводится явное различие между типами и переменными. Прежде чем описать некоторую переменную, необходимо определить некоторый тип (идентификатор этого типа), и только после этого можно описать переменную этого типа. Исключение составляют стандартные типы. Их не нужно определять в программе. Можно считать, что они являются предопределенными типами и их определения заданы в окружении программы.

Концепция типов в языке программирования Паскаль приводит к нескольким важным следствиям:

1. каждый объект имеет один и только один тип;
2. тип любого объекта можно определить, просматривая статический текст программы, нет необходимости выполнять программу, чтобы узнать тип объекта;
3. над каждым типом определен ограниченный набор операций, что позволяет статически проверить правильность их использования в каждом конкретном случае;
4. в язык вводится некоторая избыточность, которая позволяет обнаруживать ошибки.

Взаимоотношение между переменной и типом можно интерпретировать так: определение типа задает некоторый шаблон для представления объектов в памяти, который затем используется при отведении памяти для хранения переменных этого типа. Вообще говоря, в программах на языке Паскаль не обязательно определять идентификатор типа. В разделе описания переменных можно непосредственно описать тип, которому принадлежит данная переменная. Вместе с тем, синтаксис языка Паскаль требует, чтобы идентификатор типа обязательно использовался в следующих четырех случаях:

– в секции формальных параметров (за исключением согласованной схемы);

– в признаке варианта записи с вариантами;

– при определении типа указателя;

– для указания типа результата, вычисляемого с помощью функции.

Правила построения типов в языке Паскаль очень простые. Имеется несколько стандартных типов и несколько стандартных методов определения новых типов. Чаще всего новый тип определяется в терминах определенных ранее составляющих типов. Значениями этих новых типов являются структуры

данных, которые можно строить из значений составляющих типов. Составляющие типы, в свою очередь, сами могут быть разложены на компоненты (рис.1). Рассмотрим дерево типов языка программирования Паскаль. Оно построено по принципу «вырастания» одних типов из других.

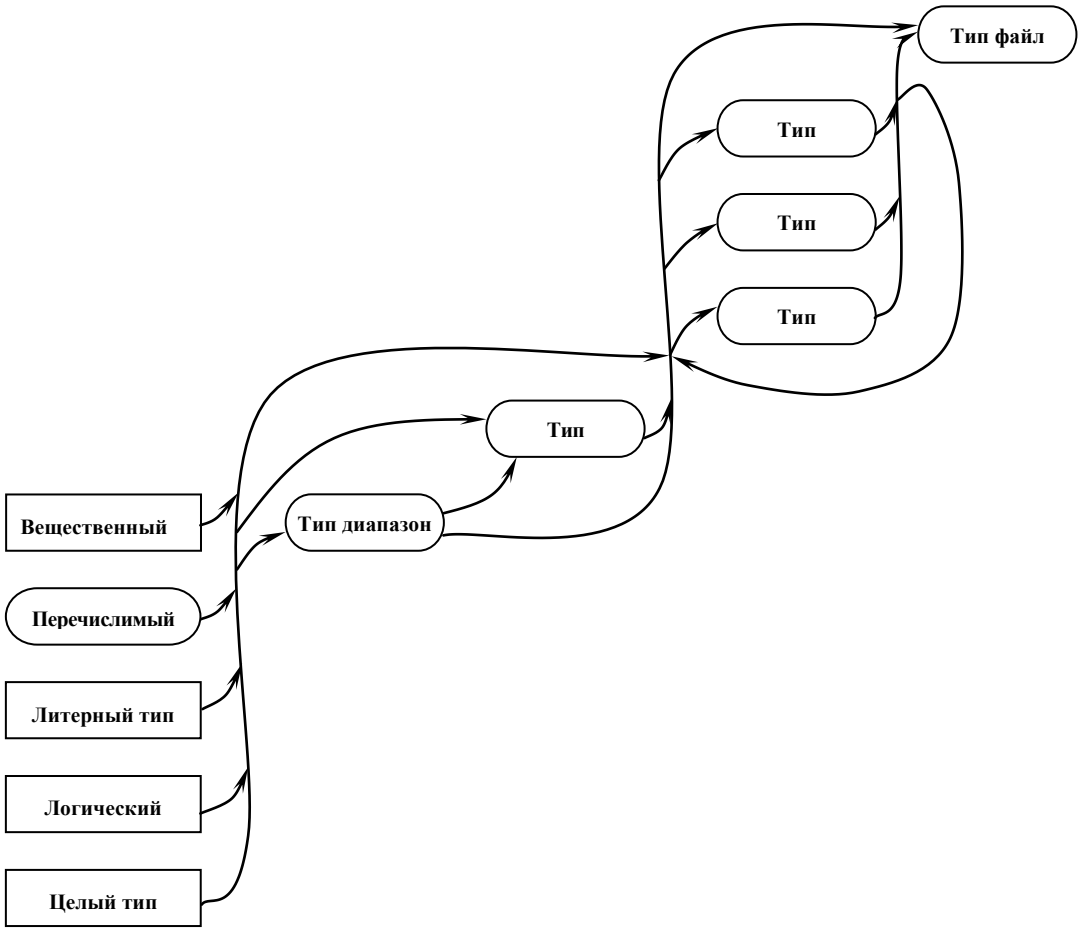


Рис.1. Классификация типов данных

Если взять любой тип, изображенный на рисунке 1, то легко определить, какие типы можно использовать при его построении: это все типы, расположенные левее и ниже и от которых идут стрелки к данному типу. Например, при построении типа множество можно использовать целый, логический, литерный, перечислимый и тип диапазон. Вещественный тип использовать нельзя – стрелка от него не идет к типу множество. Для следующих типов: массив, запись и указатель – обратим внимание на петлю, которая обозначает, что каждый из перечисленных типов можно использовать при построении этих же типов.

Все типы языка Паскаль делятся на простые и структурированные. К простым типам относятся: целый, вещественный, логический, литерный, перечислимый и тип–диапазон. Первые четыре называются стандартными типами. Перечислимый и тип–диапазон относятся к простым типам, определяемым пользователем. Все простые типы, за исключением вещественного, относятся к порядковому типу.

9.1. Целый тип

Для описания переменных целого типа используется стандартный идентификатор **INTEGER**, например:

VAR I, J, K:INTEGER;

Над элементами целого типа определены следующие операции:

– сложение (+);

– вычитание (–);

– умножение (*);

– целочисленное деление (DIV);

– остаток от целочисленного деления (MOD);

Стандартные функции дают результат целого типа:

ABS(X) – вычисляет абсолютную величину X;

SQR(X) – вычисляет $X^{2;}$

SUCC(X) – возвращает следующее целое число (X+1);

PRED(X) – возвращает предыдущее целое число (X–1);

ORD(X) – возвращает порядковый номер значения выражения X;

INC(X) – X:=X+1 – увеличивает значение X на единицу;

INC(X,N) – X:=X+N – увеличивает значение X на N;

DEC(X) – X:=X–1 – уменьшает значение X на единицу;

DEC(X,N) – X:=X–N – уменьшает значение X на N;

TRUNC(X) – если X целого типа, то остается без изменения, если x вещественного типа, результат имеет целый тип. Сохраняется целая часть X, а дробная часть теряется;

ROUND(X) – если X вещественного типа, то результат имеет целый тип, округление до ближайшего целого числа.

CHR(X) – возвращает символ по его коду;

ODD(X) – проверка целой величины на нечетность (имеет значение истина, если аргумент нечетный, и значение ложь, если аргумент четный).

9.2. Вещественный тип

Для описания переменных вещественного типа в разделе описания переменных используется стандартный идентификатор **REAL**, например:

VAR A, B : REAL;

Если хотя бы один из операндов имеет вещественный тип, то следующие операции также дают в качестве результата вещественный тип.

Над элементами вещественного типа определены следующие операции:

– сложение (+);

– вычитание (–);

– умножение (*);

– деление (/) – оба операнда могут иметь целый тип, но результат всегда вещественный.

Стандартные функции **ABS(X)** и **SQR(X),** примененные к вещественным аргументам, дают вещественный результат.

Стандартные функции **SIN(X), COS(X), ARCTAN(X), LN(X), EXP(X), SQRT(X)** независимо от типа аргумента дают вещественный тип результата.

Функция **FRAC(X)** возвращает дробную часть X, функция **INT(X)** – целую часть X.

Действия над вещественными числами всегда выполняются с машинной точностью.

9.3. Символьный тип

Для описания переменных литерного типа в разделе описания переменных используется стандартный идентификатор **CHAR**. Например,

VAR C : CHAR;

Над литерными переменными не определены никакие арифметические операции, но применимы все операции отношения (> < = <> <= >=).

Для них определены функции:

ORD(X) – определяет порядковый номер символа;

CHR(X) – по порядковому номеру символ;

UPCASE(C) – переводит строчную латинскую букву в заглавную, в противном случае возвращает сам символ;

PRED(X) – предыдущий символ;

SUCC(X) – последующий символ.

9.4. Логический тип

Определяет диапазон логических значений, который содержит два элемента: False (0) – ложь и True (1) – истина.

Для описания величин логического типа используется стандартный идентификатор **BOOLEAN**, например:

VAR L : BOOLEAN;

К переменным логического типа применимы операции отношения (> < = <> <= >=).

Для булевых переменных определены логические операции:

NOT – отрицание

AND – логическое "и" (конъюнкция)

OR – логическое "или" (дизъюнкция)

XOR – исключающее "или"

Результат применения логических операций:

Значение выражения		Результат			
X	Y	**NOT X**	**X AND Y**	**X OR Y**	**X XOR Y**
1	1	0	1	1	0
1	0	0	0	1	1
0	1	1	0	1	1
0	0	1	0	0	0

9.5. Перечислимый тип и тип–диапазон

Перечислимый тип определяется как упорядоченный набор идентификаторов, который задается путем их перечисления и описывается в разделе типов. Например,

TYPE COLOR=(black, red, green);

VAR B:COLOR;

Переменные перечислимого типа можно объявлять без предварительного описания этого типа.

VAR COLOR : (black, red, green);

Все константы упорядочены (0, 1, 2, …), и поэтому к ним можно применять функции: **PRED(X)** и **SUCC(X)**.

Можно по наименованию определить номер в списке – **ORD(X),** например, ORD(red) =1.

Тип-диапазон – это подмножество базового типа, в качестве которого может выступать любой порядковый тип. Для него задаются две константы, определяющие границы диапазона значений для данной переменной.

Тип-диапазон можно описывать в разделе типов или переменных, например,

TYPE B=1..10;

VAR A:B;

GOD:1982..2003;

CH:’A’..’Z’;

Тип-диапазон наследует все свойства своего базового типа.

Существуют две функции, поддерживающие работу с типами-диапазон:

HIGH(X) – возвращает максимальное значение типа-диапазон;

LOW(X) – возвращает минимальное значение типа-диапазон.

10. ОПЕРАТОРЫ ЯЗЫКА ПАСКАЛЬ

10.1. Оператор присваивания

переменная := выражение;

Переменная и выражение должны быть одного типа.

Исключением является случай, когда используется переменная вещественного типа, выражение – целого типа.

Например: A:=5;

C:=’B’;

Запрещено в одном выражении два оператора присваивания: A:=B:=8.6;

Пример: Вычислить значение функции $y = \frac{2-(x-3)^2}{(x-3)^2+4}$.

```
Program prim1;
Var     x: integer;
y: real;
Begin
```

```
Write(‘Введите значение x ’);
Readln (x);
y:=(2–(sqr (x–3)))/(sqr (x–3)+4);
Writeln (‘Значение функции =’, y:6:2);
Readln;
End.
```

10.2. Условный оператор

IF условие THEN оператор 1 ELSE оператор 2;

Условие может быть как простым, так и составным. В случае составного условия используются логические операции – NOT, AND, OR.

Операторы также могут быть простыми и составными. *Составной оператор* – это последовательность произвольных операторов программы, заключенная в операторные скобки **BEGIN…END**.

Например: **IF условие THEN BEGIN**

оператор 11;
оператор 12;
…
оператор 1n

Оператор 1

END
ELSE BEGIN

оператор 21;
оператор 22;
…
оператор 2n

Оператор 2

END;

Допускается укороченный вариант условного оператора:

IF условие THEN оператор;

Пример: Вычислить значение функции $y = \begin{cases} a + 6x, & x < a \\ ax + 3x^2, & x \geq a \end{cases}$

```
Program prim2;
Var x, A: integer;
y: real;
Begin
Writeln (‘Введите значение x и a’);
Readln (x, A);
If x<A then y:=A+6*x
Else y:=A*x+3*sqr(x);
Writeln (‘Значение функции =’, y:6:2);
Readln;
End.
```

10.3. Оператор выбора

CASE <переменная выбора> OF
список констант 1 : оператор 1;
список констант 2 : оператор 2;
…
список констант n : оператор n
ELSE оператор n+1
END;

Часть ELSE не является обязательной.

Пример: Составить программу, в которой даются рекомендации пешеходу при вводе первой буквы цвета светофора.

```
Program prim3;
Var k: char;
Begin
Writeln (‘Введите значение k’);
Readln (k);
Case k of
‘К’, ‘к’: Writeln (‘Стой’);
‘Ж’, ‘ж’: Writeln (‘Жди’);
‘З’, ‘з’: Writeln (‘Иди’)
else Writeln (‘Введена буква, не соответствующая цвету
светофора!’);
End;
End.
```

10.4. Операторы цикла

а) Цикл с известным числом повторений

FOR <переменная> := <выражение_1> TO <выражение_2> DO <оператор>;

если **выражение_1** <= **выражения_2**, то **оператор** будет выполняться

FOR <переменная> := <выражение_1> DOWNTO <выражение_2> DO <оператор>;

если **выражение_1** >= **выражения_2**, то **оператор** будет выполняться

Пример: Составить программу для подсчета суммы $S = \sum_{i=1}^{n} \frac{1}{i^2}$.

```
Program prim4;
Var  i, n: integer;
s: real;
Begin
Writeln (‘Введите n’);
Readln (n);
s:=0;
```

```
For i:=1 to n do s:=s+1/sqr(i);
Writeln (‘Сумма ряда = ’, s:5:2);
Readln;
End.
```

б) Цикл с предусловием

WHILE (условие) DO <оператор>;

Сначала вычисляется условие. Если условие "истина" (1), то выполняется оператор, иначе управление передается следующему за WHILE оператору. Условие должно быть подготовлено до цикла. Также необходимо предусмотреть изменение условия внутри цикла. Причем шаг может быть любым.

Пример: Составить программу для подсчета бесконечной суммы с заданной точностью (eps) $S = \sum_{i=1}^{\infty} \frac{1}{i^2}$.

```
Program prim5;
Var  i: integer;
s, eps: real;
Begin
s:=0; i:=1;
Writeln (‘Введите точность ’);
Read(eps);
While 1/sqr(i)>=eps do begin
s:=s+1/sqr(i);
i:=i+1;
End;
Writeln (‘Сумма ряда = ’, s:5:2);
Readln;
End.
```

в) Цикл с постусловием

REPEAT <оператор> UNTIL (условие);

Оператор цикла с постусловием выполняется до тех пор, пока условие не станет истинным. Условие должно быть подготовлено до цикла. Также необходимо предусмотреть изменение условия внутри цикла. Причем шаг может быть любым.

Пример: Составить программу для подсчета бесконечной суммы с заданной точностью (eps) $S = \sum_{i=1}^{\infty} \frac{1}{i^2}$.

```
Program prim6;
Var  i: integer;
s, eps: real;
Begin
s:=0; i:=1;
```

```
Writeln (‘Введите точность ’);
Read(eps);
Repeat
s:=s+1/sqr(i);
i:=i+1;
Until 1/sqr(i)<eps;
Writeln (‘Сумма ряда = ’, s:5:2);
Readln;
End.
```

11. МАССИВЫ

11.1. Вектор (одномерный массив)

Вектор – это конечное упорядоченное поименованное множество данных одного и того же типа, называемых элементами вектора.

Вектор обычно задается следующим образом:

TYPE <имя типа>= ARRAY[1..n] OF <тип элементов>;

{где n – константа, определяющая количество элементов массива }

VAR <имя массива >:<имя типа >;

Другой вариант:

VAR <имя массива>: ARRAY[1..n] OF <тип элементов>;

Обращение к элементу массива осуществляется по его индексу, например: MAS[5]:=6.83;

Ввод и вывод элементов массива – это повторяющиеся действия, поэтому работа с массивом производится с помощью цикла. При этом счетчик цикла, как правило, применяется для индексации элементов массива.

Например:

FOR I:=1 TO 9 DO READ(MAS[I]); {заполнение элементов массива}

FOR I:=1 TO 9 DO WRITE (MAS[I], ‘ ’); {вывод элементов массива}

Пример: Заполнить массив целых чисел. Найти сумму элементов массива.

```
Program prim7;
Var mas: array[1..10] of integer;
s:integer;
i:byte;
Begin
Writeln (‘Введите элементы массива’);
For i:=1 to 10 do
Read(mas[i]);
s:=0;
For i:=1 to 10 do
s:=s+mas[i];
Write (‘Сумма элементов массива=’, s);
Readln;
```

```
End.
```

11.2. Строки

Описание:

<переменная 1>:STRING[n];

{n – максимальное количество символов в строке}

<переменная 2>:STRING;

Для строковых данных применяются стандартные функции:

LENGTH(S) – возвращает текущую длину строки.

CONCAT(S1, S2, …, SN) – объединяет несколько строк.

POS(SUBSTR, S) – ищет подстроку в строке.

COPY(S, INDEX, COUNT) – возвращает подстроку, содержащуюся в строке.

Процедуры:

VAL (S, V, CODE) – преобразует строковое значение в числовое.

STR(X, S) – преобразует числовое значение в строку цифр.

INSERT(S1, S2, INDEX) – вставляет подстроку S1 в строку S2.

DELETE(S, INDEX, COUNT) – удаляет подстроку из строки.

FILLCHAR(S, COUNT, CH) – заполнение строки символами.

Пример: Дана строка символов S. Скопировать половину этой строки в S1, а вторую половину в S2.

```
Program prim8;
Var s:string;
s1, s2: string[128];
k, n: integer;
Begin
Writeln (‘Введите строку’);
Readln (s);
n:=length (s);
k:=n div 2;
s1:=copy(s, 1, k); s2:=copy(s, k+1, n–k);
Writeln(‘Полученные строки: ’);
Writeln(s1);
Writeln(s2);
Readln;
End.
```

11.3. Матрица (двумерный массив)

Матрица – это массив, у которого все элементы расположены на пересечениях строк и столбцов, или это такой вектор, каждый элемент которого есть вектор.

Массив задается следующим образом:

TYPE <имя типа>= ARRAY [1..n, 1..m] OF <тип элементов>;

{где n – количество строк массива, m – количество столбцов массива}

VAR <имя массива >:<имя типа >;

Или

VAR <имя массива>: ARRAY [1..n, 1..m] OF <тип элементов>;

Для заполнения двумерного массива необходимо иметь два цикла: внешний индексирует элементы строк, внутренний – элементы столбцов.

Например:

```
FOR I:=1 TO n DO
FOR J:=1 TO m DO READ(MAS[I, J]);
```

Пример: Заполнить двумерный массив 5×5. Поменять местами 2-й и последний столбцы массива. Измененный массив вывести на экран.

```
Program prim9;
Var m:array[1..5, 1..5] of integer;
c:integer;
i, j:byte;
Begin
Writeln('Введите элементы массива');
For i:=1 to 5 do
For j:=1 to 5 do
Read(m[i, j]);
For i:=1 to 5 do
Begin
c:=m[i, 2];
m[i, 2]:=m[i, 5];
m[i, 5]:=c;
End;
Writeln('Вывод измененного массива');
For i:=1 to 5 do
Begin    {Вывод элементов строки массива}
For j:=1 to 5 do
Write (m[i, j], '  ');
Writeln;
End;
End.
```

11.4. Сортировка массива

В словарях слово "сортировка" определяется как "распределение, отбор по сортам, деление на категории". В программировании это слово используется в гораздо более узком смысле – как сортировка предметов в возрастающем или убывающем порядке.

Было изобретено множество различных способов сортировки. Разнообразие методов объясняется тем, что каждый метод имеет свои преимущества и недостатки, поэтому он оказывается эффективнее других при некоторых конфигурациях данных и аппаратуры. В связи с этим полезно изучить характеристики каждого метода сортировки, чтобы производить разумный выбор для конкретных приложений.

Для того чтобы обоснованно сделать такой выбор, рассмотрим параметры, по которым будет производиться оценка алгоритмов:

1. *Время сортировки* – основной параметр, характеризующий быстродействие алгоритма.
2. *Память* – ряд алгоритмов требует выделения дополнительной памяти под временное хранение данных.
3. *Устойчивость* – устойчивая сортировка не меняет взаимного расположения равных элементов.
4. *Естественность поведения* – эффективность метода при обработке уже отсортированных или частично отсортированных данных. Алгоритм ведет себя естественно, если учитывает эту характеристику входной последовательности и работает лучше.

Еще одним важным свойством алгоритма является его сфера применения. Здесь основных позиций две:

- внутренние сортировки работают с данными в оперативной памяти (ОП) с произвольным доступом;
- внешние сортировки упорядочивают информацию, расположенную на внешних носителях.

Это накладывает дополнительные ограничения на алгоритм:

- доступ к внешнему носителю осуществляется последовательным образом: в каждый момент времени можно считать или записать только элемент, следующий за текущим элементом.
- объем данных большой и не позволяет разместить их в ОП.

Кроме того, доступ к данным на носителе производится намного медленнее, чем операции с оперативной памятью, выполнение действий над данными в оперативной памяти.

Внутренняя сортировка оперирует с массивами, целиком помещающимися в оперативной памяти с произвольным доступом к любой ячейке. Данные обычно сортируются на том же месте, без дополнительных затрат памяти.

Внешняя сортировка оперирует с запоминающими устройствами большого объема, но с доступом не произвольным, а последовательным (сортировка файлов), т.е. в данный момент мы 'видим' только один элемент, а затраты на перемотку по сравнению с памятью неоправданно велики. Это приводит к специальным методам сортировки, работающим в условиях ограниченного доступа.

Виды внутренней сортировки (Дональд Кнут. Искусство программирования, том 3. Сортировка и Поиск. http://rutracker.org/forum/viewtopic.php?t=3063744):

Сортировка вставками – предполагается, что перед рассмотрением последнего элемента все предыдущие упорядочены, и этот элемент вставляется в соответствующее место.

a. Простые вставки
b. Бинарные вставки или двухпутевые вставки

c. Метод Шелла
d. Вставки в список
e. Сортировка с вычислением адреса

2. *Обменная сортировка* – предусматривает обмен местами между парами элементов, в которых нарушается упорядоченность, до тех пор, пока таких пар не останется.

a. Метод "Пузырька" (обменная сортировка с выбором)
b. Параллельный метод Бэтчера (обменная сортировка со слиянием)
c. Быстрая сортировка Хоара (обменная сортировка с разделением)
d. Поразрядная обменная сортировка

3. *Сортировка посредством выбора* – основана на идее многократного выбора.

a. Сортировка посредством простого выбора
b. Выбор из дерева
c. Пирамидальная сортировка

4. *Сортировка слиянием* – заключается в объединении двух или более упорядоченных массивов в один упорядоченный массив. При этом сравниваются два наименьших элемента, выводится наименьший из них, и процедура повторяется.

5. *Распределяющая сортировка* – (про карты)

Виды внешней сортировки:

1. Многопутевое слияние и выбор с замещением
2. Многофазное слияние
3. Каскадное слияние
4. Чтение ленты в обратном направлении
5. Осциллирующая сортировка
6. Внешняя поразрядная сортировка
7. Сортировка с двумя лентами

12. ЗАПИСИ

12.1. Тип запись

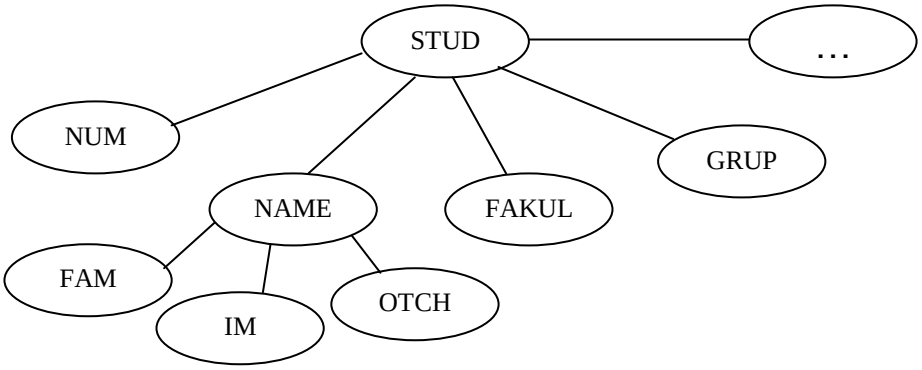
Запись – это конечное упорядоченное множество элементов различных типов данных, объединенных под одним именем.

Доступ к любому элементу осуществляется с помощью имени элемента, который дан при описании логической структуры, например:

STUD.FAKUL:=’ТФ’;

Для простой записи характерно, что каждый элемент является простым данным или цепочкой простых данных.

Если отобразить логическую структуру в виде записи, то ее корнем будет имя записи, а листьями – непосредственно элементы записи.



Общим видом записи является такая запись, в которой каждый ее элемент может быть, в свою очередь, записью более низкого уровня.

Независимо от числа уровней в структуре записи, полезная информация содержится лишь в тех элементах, которые соответствуют листьям.

Доступ к содержательному элементу осуществляется указанием последовательности имен в промежуточных элементах и соответствующего содержательного элемента, например: STUD.NAME.FAM:=’Иванов’;

Общая структура записи:

```
TYPE <имя типа>=RECORD
        <идентификатор поля>:<тип>;
        …
    END;
```

12.2. Оператор присоединения имен

Структура:

```
WITH <имя записи> DO
    BEGIN
        <идентификатор поля> := значение;
        …
    END;
```

При применении оператора WITH не разрешается иметь в разных записях одинаковые имена полей.

Пример: Составить программу ввода данных об N студентах (ФИО и номер группы). Вывести список студентов интересующей группы.

```
Program prim10;
Type stud=Record
        fio: string[20];
        group: string[7];
End;
Var x: array[1..100] of stud;
        i, kol, n: byte;
        gr: string[7];
Begin
```

```
Write(‘Введите количество студентов: ’);
Readln (n);
For i:=1 to n do with x[i] do
Begin
        Writeln(‘Введите данные ‘,i,’–го студента’);
        Write(‘Введите ФИО: ’);
        Readln(fio);
        Write(‘Введите номер группы: ’);
        Readln(group);
End;
Writeln;
Write(‘Введите номер интересующей группы: ’);
Readln(gr);
kol:=0;
For i:=1 to n do with x[i] do
  If group=gr then
  Begin
        kol:= kol+1;
        if  kol=1 then Writeln(‘Список студентов группы ’, gr);
        Writeln(kol, ‘.  ’, fio);
  End;
If kol=0 then
  Writeln(‘Данные о студентах группы ’, gr, ‘ отсутствуют’);
Readln;
End.
```

12.3. Запись с вариантами

Структура:

```
TYPE <имя типа>=RECORD
                    <идентификатор поля>:<тип>;
                    …
                    CASE <признак варианта>:<тип> OF
                          <константа>:(<идентификатор поля>:<тип>;)
                    …
END;
```

Пример: Составить программу для ввода данных об абитуриентах.

```
        Program prim11;
        Uses crt;
        Type stud=record
              Fio :string[25];
              Adr: string[18];
              Case pol: char of
                    'м':(voen:string[15]);
                    'ж':();
```

```
End;
Var mas: array[1..10] of stud;
        i, n, pol_pr : byte;
Begin
Clrscr;
Write('Введите количество записей: ');
Readln(n);
For i:=1 to n do with mas[i] do begin
        Write('Введите фамилию: '); Readln(fio);
        Write('Введите адрес: '); Readln(adr);
        Write('Введите признак пола(0-м, 1-ж): '); Readln(pol_pr);
        Case pol_pr of
                0:pol:='м';
                1:pol:='ж';
        End;
        Case pol of
        'м': begin
                Write('Введите N приписного(военного билета) ');
                Readln(voen);
        End;
        'ж':;
        End;
End;
For i:=1 to n do with mas[i] do
        Writeln(fio:15, adr:15, pol:2, voen:8);
        Readln;
End.
```

13. МНОЖЕСТВА

Множество в математике – это произвольный набор объектов любой природы, понимаемых как единое целое. На вид объектов и их число не накладывается никаких ограничений. В Паскале понятие множества несколько уже.

Множество – это структурированный тип данных, представляющий собой множество всех подмножеств первоначального (базового) множества, включая пустое множество.

Два множества, отличающиеся только порядком следования элементов, называются одинаковыми. Следовательно, элементы множества не упорядочены, поэтому одинаковы следующие множества: {A,B,C}, {A,C,B}, {B,C,A} и т.д.

В теории множеств нет никаких ограничений на типы элементов, которые могут быть членами множества. В языке Паскаль допускается только

ограниченное число типов для элементов множества. Это может быть любой простой тип со следующими ограничениями:

– вещественный тип использовать нельзя;

– для целого типа можно задавать только диапазоны.

– последнее ограничение связано с числом элементов во множестве. Оно определяется конкретной реализацией и может принимать значения от 0 до 255.

Такая зависимость типа-множество от ЭВМ приводит к потере переносимости программ. Обычно множества представляются в виде строки бит, занимающих одно или несколько машинных слов. Обработка битовых строк производится очень быстро, поэтому все операции над множествами также выполняются очень быстро.

В общем виде множественный тип данных задается следующим образом:

TYPE <имя типа> = SET OF <базовый тип>;

<базовый тип> определяет конечное множество значений, которые образуются перечислением через запятую элементов базового множества или диапазоном. Множество, не содержащее элементов, называется пустым и обозначается [ ].

Пример описания переменных множественных типов:

```
TYPE TOWN=(MOSKOW, KIEV, RUBTSOVSK);
VAR P1, PP1:SET OF TOWN;
    P2, PP2:SET OF 0..9;
    P3, PP3:SET OF 'A'..'F';
```

Переменная P1 может принимать значения:

[MOSKOW, KIEV, RUBTSOVSK] []

[MOSKOW, KIEV] [MOSKOW, RUBTSOVSK] [KIEV, RUBTSOVSK]

[MOSKOW] [KIEV] [RUBTSOVSK]

В общем случае, если базовое множество содержит N элементов, производный множественный тип определяет 2^N подмножеств.

Принятая в ЭВМ двоичная форма хранения информации позволяет эффективно реализовать представление множеств и операции над ними. Для размещения в памяти значений каждой переменной множественного типа TOWN достаточно трех битов. Пример:

P1:=[KIEV] PP1:=[MOSKOW, RUBTSOVSK]

P1

0	**0**	**1**
MOSKOW	KIEV	RUBTSOVSK

PP1

1	**0**	**1**
MOSKOW	KIEV	RUBTSOVSK

Если элементы множества являются последовательными значениями базового множества, то можно указывать только первый и последний элементы.

Пример (записи идентичны): P3:=[‘A’..’F’];

PP3:=[‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’];

Множество [‘C’..’A’] является пустым, так как в перечислении объектов базового типа порядковый номер ‘C’ больше порядкового номера ’A’.

Элементы множеств могут задаваться и выражениями соответствующего базового скалярного типа. Например:

k:=7; p2:=[1, k+1]; pp2:=[k-1..8];

множество p2 состоит из двух элементов – [1, 8]

множество pp2 состоит из трех элементов – [6, 7, 8]

Над множествами определены три арифметические операции:

(+) – объединение

(*) – пересечение

(–) – разность

С их помощью можно строить различные выражения множественного типа.

Объединением двух множеств называется множество элементов, принадлежащих обоим множествам.

Например: [‘B’, ’F’]+[‘B’..’D’]=[‘B’, ’C’, ’D’, ’F’]

Пересечением двух множеств называется множество тех элементов, которые принадлежат одновременно двум множествам.

Например: [0..4]*[5, 6]=[ ]

[Киев, Ялта]*[Москва, Киев]=[Киев].

Разностью двух множеств называется множество, содержащее элементы первого множества, которые не являются элементами второго множества.

Например: [1, 5, 9]–[2, 4, 8, 9]=[1, 5].

Для сравнения множеств используются операции отношения: = <> <= >=.

Выражение **A=B** истинно только тогда, когда сравниваемые множества содержат одни и те же элементы.

Выражение **A<>B** истинно только тогда, когда одно из сравниваемых множеств содержит хотя бы один элемент, не входящий в другое множество.

Выражение **A<=B** истинно только тогда, когда все элементы множества **A** одновременно являются элементами множества **B**.

Выражение **A=>B** истинно только тогда, когда все элементы множества **B** одновременно являются элементами множества **A**.

Существует операция принадлежности элемента некоторому множеству – **IN**. Например: 5 IN [3..7] имеет значение true.

Приоритет выполнения операций следующий: * + – IN = <> <= >=. Для изменения порядка используются круглые скобки.

Средства работы со множествами позволяют в некоторых случаях сократить программы и сделать их более наглядными и эффективными за счет уменьшения числа различных проверок.

Для вывода элементов некоторого множества, сформированного в процессе выполнения программы, необходимо воспользоваться оператором

цикла, внутри которого была бы проверка на принадлежность текущего значения параметра цикла выводимому множеству.

Например:

```
VAR  B:SET OF 1..9;
     I:INTEGER;
BEGIN
B:=[1..5];
FOR I:=0 TO 9 DO
     IF I IN B THEN WRITE(I:3);
END.
```

14. ПОДПРОГРАММЫ

14.1. Процедуры и функции

Подпрограмма – это набор операторов, выполняющих определенное, законченное по смыслу действие, оформленное специальным образом.

В Паскале существует два вида подпрограмм – *процедуры* и *функции*.

Функция – это подпрограмма, имеющая один (и только один) выходной параметр. Т.е. она всегда возвращает в точку вызова скалярное значение, которое нужно принять и разместить.

Синтаксис описания функции:

FUNCTION <имя> (<имя входного параметра>:<тип >, …): <тип выходного параметра>;
{раздел описаний}
BEGIN
{тело функции}
<имя>:=<простые функции>;
END;

{раздел описаний} – аналогично разделу описаний в программе (метки, типы, константы,…).

{тело функции} – набор операторов Паскаль. Последним из операторов должен быть оператор присваивания имени функции ее значения. Через имя функции передается результат работы функции (выходной параметр).

Входные параметры – это ноль или более переменных, например:

Function ex1(a, b:real; c:char):real;
Function ex2:real;

Вызов функции:

<имя переменной>:=<имя функции[(входные параметры)]>;

Пример: Написать функцию, которая сравнивает два вещественных числа и возвращает результат сравнения в виде одного из знаков < > =

```
Program prim12;
Var a,b: real;
Function srav(a1,b1:real):char;
Begin
```

```
        If a1<b1 then srav:='<' else
        If a1>b1 then srav:='>' else srav:='=';
End;
Begin
Write ('Введите два вещественных числа: ');
Readln (a, b);
Writeln ('Результат сравнения ', a:6:2, srav (a,b), b:6:2 );
End.
```

Процедура – аналогична функции, но может возвращать ноль и более параметров.

Синтаксис описания процедуры:

PROCEDURE <имя> [(<имя входного параметра>:<тип >, …; VAR <имя выходного параметра>:<тип >, …)];

{раздел описаний}

BEGIN

{тело процедуры}

END;

Вызов процедуры:

<имя процедуры [(входные параметры, выходные параметры)]>;

Пример: Написать процедуру, которая формирует строку, состоящую из звездочек. Длина строки является входным параметром процедуры. В головной программе вывести строку на экран.

```
Program prim13;
Var   n:byte;
      len:string;
Procedure Lens (n1:integer; Var len1:string);
Var   i:byte;
      ch:char;
Begin
      ch:='';
      For i:=1 to n1 do
      Fillchar(len1, n, ch);
End;
Begin
Write ('Введите количество звездочек: ');
Readln (n);
lens(n, len)
Writeln('Полученная строка – ', len);
Readln;
End.
```

14.2. Стандартные процедуры и функции

BREAK – обеспечивает выход из операторов повторения. Управление передается следующему оператору.

CONTINUE – завершение очередной итерации оператора повторения. Управление передается на начало оператора.

EXIT – выход из текущей подпрограммы. При вызове из тела программы завершает ее работу.

HALT [(CODE:Word)] – останавливает выполнение программы и возвращает управление в операционную систему. **CODE** определяет код завершения программы.

GETDATE(Var year, month, day, dayofweek:word) – возвращает текущую машинную дату. Применяется в отчетах на текущую дату.

GETTIME(Var hour, min, sec, sec100:word) – возвращает текущее машинное время. Применяется в документах, где время играет важную роль, например в экспериментах, программах реального времени.

SETDATE(year, month, day:word) – устанавливает текущую машинную дату.

SETTIME(hour, min, sec, sec100:word) – устанавливает текущее машинное время.

15. ВНЕШНИЕ ПРОЦЕДУРЫ И МОДУЛИ

Если процедура транслируется отдельным модулем, хранится на диске отдельно от программ, использующих ее, то она называется *внешней*.

В головной программе для этой процедуры указывается ее заголовок и служебное слово **EXTERNAL**. Например,

```
PROCEDURE EXT (A:INTEGER; VAR B:REAL);
EXTERNAL;
```

В тексте программы при объявлении внешних подпрограмм нужно задать директиву компилятору $L, аргументом которой является имя **OBJ**-файла, содержащего код подключаемой программы.

Например: **PROGRAM PR;**
{$L M1.OBJ}

Иногда процедуру необходимо вызвать раньше, чем она будет описана. Заголовок такой процедуры описывается с помощью служебного слова **FORWARD**.

```
PROCEDURE EXT (A:INTEGER; VAR B:REAL);
FORWARD;
```

Другой тип внешних процедур – процедура типа блок – **UNIT.**

UNIT используется для создания собственных и стандартных библиотек подпрограмм. Оформление происходит по следующим правилам.

```
UNIT <имя>;
INTERFACE
<секция связи>
IMPLEMENTATION
<секция реализации>
```

BEGIN
<секция инициализации>
END.

В секции **INTERFACE** описываются все внешние объявления (типы, константы, переменные, подпрограммы с описанием параметров), которые должны стать доступными основной программе.

Секция **IMPLEMENTATION** содержит описание локальных типов, констант, переменных, подпрограмм, объявленных в секции связи, а при необходимости и других.

В секции инициализации размещаются исполняемые операторы.

Если программа использует несколько UNIT, то их части инициализации будут выполняться в порядке их перечисления в разделе USES.

В головной программе все созданные UNIT подключаются в разделе USES.

В системе программирования Паскаль используется 3 типа файлов:

1. Текстовые файлы, содержащие программы на Паскале с расширением .PAS.

2. .TPU-файлы.

3. .EXE и файлы, готовые к запуску.

.TPU-файлы создаются в результате трансляции процедур Unit и являются аналогом объектного файла (.OBJ). В отличие от .OBJ .TPU-файлы содержат информацию о типах формальных параметров и результатах функций, что позволяет отнести их к внешним процедурам.

Отличие .TPU-файлов от .OBJ дает возможность на этапе трансляции проводить детальный анализ корректности обращения к процедурам и функциям.

Введение .TPU-файлов позволяет создавать программы (.EXE файлы), превышающие 64Кб. Однако каждый .TPU-файл не может превышать 64Кб.

Следует помнить, что глобальные переменные и типизированные константы из всех Unit собираются в один сегмент данных, который не может быть >64Кб. Это ограничение напоминает о том, что глобальных переменных и типизированных констант не может быть много и что одной из ошибок может быть превышение размера сегмента данных.

.TPU-файлы создаются из текстовых файлов, содержащих Unit, значит, Unit создается так же и там же, где Паскаль-программа.

При построении .EXE файла каждый Unit включается в тело программы всего 1 раз.

Замечание: Если внешняя процедура работает с массивом, то тип массив должен быть описан как глобальный в разделе **INTERFACE.**

16. ФАЙЛЫ

Файл – поименованная последовательность компонент одной структуры, расположенных на некотором устройстве во внешней памяти или иногда в ОП. Компоненты файлов могут быть любого типа, кроме типа файлов.

Файлы на диске называются *внешними*, в ОП – *внутренними* или временными. Внешние файлы могут быть последовательного и прямого доступа. *Последовательный доступ* – это возможность последовательного просмотра файла. *Прямой доступ* – это доступ к компоненте файла, осуществляющийся по № записи или по адресу. Файлом прямого доступа может быть только двоичный файл.

16.1. Текстовые файлы

Текстовые файлы – это файлы, состоящие из символов, разделенных на строки. Т.е. элементом файла является один символ. Каждая строка заканчивается символом конца строки. Весь файл заканчивается символом конца файла.

Описание текстового файла в программе следующее:

VAR <имя файловой переменной>:TEXT;

Файловые переменные используются для доступа к файлу. Файловые переменные являются указателем на системный буфер, куда помещается запись при чтении внешнего файла или откуда передается запись при создании файла на диске. Поэтому дополнительно описывается переменная для доступа к полям записи – буферная переменная. Она должна быть такого же типа, что и элемент файла. Например:

```
TYPE STUD=RECORD
FIO:STRING[20];
VOSR:BYTE;
END;
VAR F:TEXT;
BUF:STUD;
```

Процедуры и функции для работы с текстовыми файлами:

EOF(<файловая переменная>) – признак конца файла.

EOLN(<файловая переменная>) – признак конца строки.

ASSIGN(<файловая переменная>,'<имя файла на диске>') – назначение файловой переменной для конкретного файла на диске.

RESET(<файловая переменная>) – открытие файла на чтение.

REWRITE(<файловая переменная>) – открыть файл для записи.

READ [LN] (<файловая переменная>,<список переменных>) – считывание данных из файла.

WRITE [LN] (<файловая переменная>,<список переменных>) – записывает в файл данные согласно списку.

APPEND(<файловая переменная>) – открытие файла для добавления в конец новых элементов.

CLOSE(<файловая переменная) – закрытие файла.
ERASE(<файловая переменная >) – удаляет закрытый файл.
RENAME(<файловая переменная >, '<новое имя файла>') – переименовывает закрытый файл.

Пример: Дан текстовый файл целых чисел. Подсчитать сумму отрицательных элементов в файле.

```
Program prim14;
Var f:text;
s, p:integer;
Begin
Assign (f, 'Sum.txt');
Reset(f);
s:=0;
While not eof (f) do
Begin
        Read (f, p);
        If p<0 then s:=s+p;
End;
Writeln ('Сумма отрицательных элементов = ', s);
Close (f);
Readln;
End.
```

16.2. Двоичные файлы

Двоичный файл – рассматривается как последовательность записей определенной структуры. Единицей измерения такого набора данных является сама запись.

Описание двоичного файла в программе следующее:

VAR <имя файловой переменной> : FILE OF <тип элементов файла>;

Процедуры для работы с двоичными файлами:

ASSIGN(<файловая переменная>, '<имя файла на диске>')
RESET(<файловая переменная>)
REWRITE(<файловая переменная>)
CLOSE(<файловая переменная)
READ(<файловая переменная>,<переменная>)
WRITE(<файловая переменная>,<переменная>)
SEEK(<файловая переменная>,<номер записи>) – устанавливает указатель на запись с заданным номером. Отсчет номера записи ведется с 0.
ERASE(<файловая переменная >)
RENAME(<файловая переменная >,'<новое имя файла>')

Функции:

EOF(<файловая переменная>)
FILEPOS(<файловая переменная>) – позволяет определить текущую позицию файла;

FILESIZE(<файловая переменная>) – позволяет определить количество компонент в файле.

Пример: Составить программу, в которой выводятся на терминал все четные числа, записанные в двоичный файл.

```
Program prim15;
Var    f: file of integer;
       n: real;
Begin
Assign (f, 'file.dat');
Reset (f);
While not eof(f) do
Begin
       Read (f, n);
       If n mod 2 = 0 then Write (n, '  ');
End;
Close(f);
Readln;
End.
```

16.3. Файлы без типа (нетипизированные)

Файл без типа – это файл, который состоит из компонент одинакового размера, структура которых не известна или не имеет значения.

Такие файлы применяются или в процедурах копирования, или при обработке файлов базы данных.

Описание файла без типа:

VAR <имя файловой переменной> : FILE;

Нетипизированный файл является файлом прямого доступа.

Для обработки файлов без типа существуют следующие процедуры:

REWRITE(<файловая переменная>[;<размер>]) – создает новый файл.

RESET(<файловая переменная>[;<размер>]) – открывает существующий файл.

BLOCKREAD(<ф.п.>; <буф. п.>; <кол. зап.> [;<реальное кол. проч. зап.>]) – читает из файла заданное количество записей в буферную переменную.

BLOCKWRITE(<ф.п.>; <буф. п.>; <кол. зап.> [;<реальное кол. запис. зап.>]) – записывает в файл заданное количество записей из буферной переменной.

SEEK(<файловая переменная>,<номер записи>)

Для работы с файлами без типа применяются те же функции, что и для файлов с типами.

EOF(<файловая переменная>) – признак конца файла.

FILESIZE(<файловая переменная>) – позволяет определить количество компонент в файле

FILEPOS(<файловая переменная>) – позволяет определить текущую позицию файла

Пример: Копирование файла

```
Program prim16;
Var    FromF, ToF: file;
       NumRead, NumWritten: Word;
       Buf: array[1..2048] of Char;
Begin
Assign(FromF, ParamStr(1));
Reset(FromF, 1);
Assign(ToF, ParamStr(2));
Rewrite(ToF, 1);
Writeln('Copying', FileSize(FromF), 'bytes...');
Repeat
       BlockRead(FromF, Buf, SizeOf(Buf), NumRead);
       BlockWrite(ToF, Buf, NumRead, NumWritten);
Until (NumRead = 0) or (NumWritten <> NumRead);
Close(FromF);
Close(ToF);
Readln;
End.
```

17. УКАЗАТЕЛИ. СПИСКОВАЯ СТРУКТУРА

17.1. Тип указатель

Для организации динамической памяти применяются переменные, называемые указателями. Указатель в качестве своего значения содержит адрес переменой заранее определенного типа. Если указатель связан с некоторым типом данных, то он называется *типизированным*.

Объявление типизированного указателя:

TYPE <идентификатор>=^<тип переменной>;

VAR <имя 1>:<идентификатор>;

или

VAR <имя 2>:^<тип переменной>;

Можно объявлять указатель и не связывать его с каким-либо конкретным типом. Указатель, не связанный с конкретным типом данных, называется *нетипизированным*.

Объявление нетипизированного указателя:

TYPE <идентификатор> = POINTER;

VAR <имя 1>:<идентификатор>;

или

VAR <имя 2>:POINTER;

Обнулить указатель в начале или в конце работы можно с помощью функции **NIL**.

Например: **<имя переменной>:= NIL;**

MARK(<имя переменной>) – записывает в указатель текущее состояние кучи, т.е. текущий указатель свободной памяти.
RELEASE(<имя переменной>) – очищает кучу до этого адреса.

17.2. Организация списковой структуры

Часто динамические структуры физически представляются в форме *списков,* их еще называют списковыми структурами. Причем под списком подразумевается связный список.

Связный список – это такая структура, элементами которой служат записи с одним и тем же форматом, связанные друг с другом с помощью указателей, хранящихся в самих элементах.

Пример описания списковой структуры:

```
TYPE PTR=^ST;
     ST=RECORD
     V:INTEGER;
     P:PTR;
END;
VAR  X:INTEGER;
     P1, P2:PTR;
```

Организация работы со списковыми структурами может быть представлена в виде двух процедур: запись в стек и извлечение из стека.

Вход в списковую структуру возможен по указателю. Поэтому должен быть описан указатель на запись. Перед началом работы со списковой структурой нужно обнулить указатель:

```
P2:=NIL;
```

Для занесения данных в стек необходимо выполнить действия:

```
NEW(P1);
P1^.V:=X;
P1^.P:=P2;
P2:=P1;
```

Для извлечения данных из стека выполняются обратные действия:

```
P1:=P2;
X:= P1^.V;
P2:=P1^.P;
```

Дальнейшее построение списковой структуры осуществляется с помощью цикла.

17.3. Функции и процедуры для работы с динамическими переменными, указателями и адресами

NEW(<имя переменной>) – создание динамической переменной.
DISPOSE(<имя переменной>) – уничтожение динамической переменной.
ADDR(<переменная>):Pointer – возвращает адрес заданного объекта.
OFS(<переменная>):Word – возвращает смещение заданного объекта относительно начала кучи.

SPTR – возвращает текущее значение регистра SP.
MemAvail – возвращает количество имеющихся в куче свободных байт памяти.
MaxAvail – возвращает номер наибольшего непрерывного свободного блока кучи, соответствующий размеру наибольшей динамической переменной, которая может быть помещена в кучу.
FreeMem(Var <переменная>; <размер>) – уничтожает динамическую переменную указанного размера.
GetMem(Var <переменная>; <размер>) – создает новую динамическую переменную заданного размера.

Пример: С терминала ввести данные о товарах: наименование товара и цена. На основе этих данных создать списковую структуру. Организовать вывод списковой структуры на терминал.

```
Program prim17;
Type p=^tov;
        tov=record
        name:string[30];
        cen:real;
        uc:p;
End;
Var     p1,p2:p;
        b:tov;
        n, i: word;
Begin
Write('Введите количество записей');
Readln (n);
p1:=Nil;
For i:=1 to n do Begin
        Write('Введите наименование товара – ');
        Readln (b.name);
        Write('Введите цену товара – ');
        Readln (b.cen);
        New(p2);
        p2^.name:=b.name;
        p2^.cen:=b.cen;
        p2^.uc:=p1;
        p1:=p2;
End;
While p2<>nil do Begin
        Writeln (p2^.name:30, p2^.cen:8:2);
        p1:=p2^.uc;
        Dispose (p2);
        p2:=p1;
End;
```

```
Readln;
End.
```

18. ГРАФИКА

18.1. Назначение графического режима и его инсталляция

Графический режим предназначен для выполнения следующих графических функций:

1. Установка курсора и определение его позиции.
2. Установка цвета фона и линий, а также стиля заполнения.
3. Рисование геометрических фигур.
4. Работа с текстом в текстовом и графическом режимах.
5. Сохранение и восстановление изображений.

Графические функции содержатся в специальных модулях (Graph, Graph3) и подключаются в разделе USES.

Одним из понятий графики является разрешение экрана.

Разрешение экрана – это количество доступных точек или элементов изображения, состоянием которых можно управлять.

Разрешение экрана определяется числом точек (пикселей) на экране по горизонтали и вертикали. С помощью этих точек создается изображение. В зависимости от типа дисплея количество точек по горизонтали и вертикали может быть различным (640×200, 640×350...). Вся управляющая информация о точках хранится в терминальной памяти, и под нее отводится 16 Кб. В графическом режиме здесь хранится информация о 128000 точек экрана. В каждом байте – информация о 4 точках. В текстовом режиме в каждом байте информация об одном символе.

Для числовой адресации любой точки необходима система координат. В графическом Паскале есть абсолютные (рис.2) и относительные координаты.

При *абсолютных координатах* за точку (0,0) принят верхний левый угол экрана. Абсолютная координата определяется:

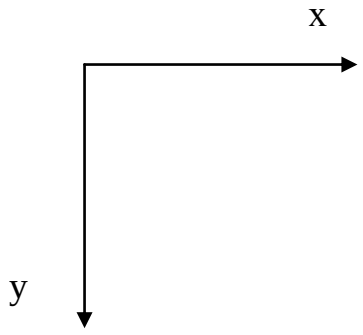


Рис. 2. Абсолютная система координат

Относительные координаты определяются относительно заданной точки на экране. При работе с относительными координатами система сама вычисляет абсолютные координаты. Относительные координаты могут быть положительными и отрицательными.

Перед началом работы в графическом режиме необходимо:

1. Подключить библиотеку графических процедур.
2. Установить графический режим.

Инсталляция графического режима осуществляется с помощью процедуры:

INITGRAPH(Gd, Gm, Path);

Gd – GraphDriver – в этой переменной задается номер драйвера или константа **Detect.**

Gm – GraphMode – это переменная, куда устанавливается режим работы драйвера.

Path – DriverPath – путь поиска драйвера. Если драйвер находится в текущем каталоге, то можно задать пробел.

При выполнении INITGRAPH результат выполнения содержится в функции **GraphResult.**

0 – нормальное завершение установки графического режима.

–1 – ошибка при установке графического режима.

Обычно с помощью этой функции осуществляется проверка работы по установке графического режима. После обращения к этой функции признак ошибки сбрасывается, поэтому повторное обращение к ней вернет 0.

GraphErrorMsg(<код>:Integer):string – по коду ошибки выдает соответствующее текстовое сообщение.

Пример: Построить окружность радиусом 100 голубого цвета в центре экрана.

```
Program prim18;
Uses graph;
Var gd, gm, err, x, y:integer;
Begin
        Gd:=detect;
        Initgraph(gd, gm, ‘ ‘);
        Err:= graphresult;
        If (err<>grok) then begin
                Writeln(grapherrormsg(err));
                Halt(1);End;
x:=getmaxx div 2;y:=getmaxy div 2;
Setcolor(2);
Circle(x, y, 100);
Closegraph;
Readln;
End.
```

GetGraphMode – возвращает значение, в котором содержится код установленного режима работы графического адаптера.

SetGraphMode(<режим> : integer) – устанавливает новый графический режим работы адаптера.

CloseGraph – завершает работу адаптера в графическом режиме и восстанавливает текстовый режим работы экрана.

В программе возможна работа в графическом и текстовом режимах. Для подключения текстового режима в разделе USES нужно указать модуль работы с окнами CRT.

TextMode(<режим> : word) – устанавливает текстовый режим работы адаптера.

RestoreCRTMode – служит для кратковременного возврата в текстовый режим. Установленные параметры графического режима не сбрасываются, и память, выделенная для размещения графического драйвера, не освобождается.

18.2. Установка курсора

Графический курсор можно устанавливать в абсолютных и относительных координатах. Сам графический курсор не виден.

GetMaxX и **GetMaxY –** возвращают значения типа **Word,** содержащие максимальные координаты экрана по горизонтали и вертикали в текущем режиме работы.

GetX и **GetY –** возвращают значения, содержащие текущие координаты указателя по горизонтали и вертикали. Координаты определяются относительно левого верхнего угла окна или, если окно не установлено, экрана.

MoveTo(x, y : integer) – устанавливает новое текущее положение указателя.

MoveRel(Dx, Dy : integer) – устанавливает новое текущее положение указателя в относительных координатах.

Например:

```
MoveTo(10, 10); {(10, 10)}
MoveRel(10, 10); {(20,20)}
```

В графическом режиме оператор вывода не работает и для вывода текста необходимо переключиться на текстовый режим.

GotoXY(x, y) – установка курсора в текстовом режиме.

В текстовом режиме возможен ввод букв в различных шрифтах, которые определяются фонтом. Есть набор файлов различных фонтов, их тип **.CHR.**

Для обозначения фонтов применяются константы:

Константа	Код	Вид
DEFAULTFONT	0	8×8 (пикселевый по умолчанию)
TRIPLEXFONT	1	тройной штриховой
SMALLFONT	2	малые буквы
SANSSERIFFONT	3	гротесковый
GOTICFONT	4	готический

Характеристикой шрифтов является направление рисования.

HorizDir – горизонтальный

VertDir – вертикальный снизу вверх и размер букв

1-8×8 нормальный шрифт

2 – 16×16 шрифт, увеличенный в два раза

…

10

Для вывода текста в графическом режиме используются процедуры:

SetTextStyle (<шрифт>, <стиль>, <размер текста>) – устанавливает текущий шрифт, стиль и размер текста.

OutText (<строка>) – выводит текстовую строку на экран.

OutTextXY (<X>, <Y>, <строка>) – выводит текстовую строку в заданные координаты экрана.

Для очистки экрана применяются процедуры:

ClearDevice – очищает графический экран. При обращении к процедуре указатель устанавливается в левый верхний угол экрана, а сам экран заполняется цветом фона.

ClearViewPort – очищает графическое окно, а если окно не определено к этому моменту, весь экран. При очистке окно заполняется цветом с номером 0 из текущей палитры. Указатель перемещается в верхний левый угол окна.

18.3. Процедуры и функции установки цвета, стиля, шаблона

SetBkColor(<цвет>:word) – установка текущего цвета фона.

SetColor(<цвет>:word) – установка текущего цвета для выводимых линий и символов.

В палитре 16 цветов с соответствующими константами:

Black	0	черный	DarkGray	8	темно–серый
Blue	1	голубой	LightBlue	9	светло–голубой
Green	2	зеленый	LightGreen	10	светло–зеленый
Cyan	3	бирюзовый	LightCyan	11	светло–бирюзовый
Red	4	красный	LightRed	12	светло–красный
Magenta	5	фиолетовый	LightMagenta	13	светло–малиновый
Brown	6	коричневый	Yellow	14	желтый
LightGray	7	светло–серый	White	15	белый

GetBkColor:word – возвращает код текущего цвета фона.

GetColor : word – возвращает код текущего цвета линий.

FloodFill (x, y, Border) – окраска произвольной замкнутой области. **X, Y** – координаты точки, принадлежащей области. **Border** – цвет границы замкнутой области. Если фигура не замкнута, закраска разольется по всему экрану.

SetFillStyle (Pattern:word, Color:word) – устанавливает стиль заполнения.

Pattern:

0 окраска фоном (узор отсутствует)

1 сплошной цвет

2 заполнение пунктиром
3 заполнение штриховкой ////
4 заполнение толстой штриховкой ////
5 заполнение толстой штриховкой \\\\
6 заполнение штриховкой \\\\
7 заполнение + + + + + + + +
8 заполнение редкими линиями ××××
9 заполнение частыми линиями ××××
10 заполнение редкими точками
11 заполнение частыми точками
12 заполнение шаблоном пользователя

SetLineStyle (LStyle, Pattern, Thick : word) – устанавливает новый стиль вычерчиваемых линий

LStyle – тип линии: 0 – сплошная линия
1 – линия из точек . . . . . .
2 – штрих-пунктир _ . _ . _ . _
3 – пунктир – – – – – – –
4 – тип пользователя

Thick – толщина линии: 1 – нормальная.
3 – утроенная толщина.

18.4. Процедуры и функции построения фигур

PutPixel (X, Y : integer, Color : word) – выводит точку заданным цветом в указанные координаты.

GetPixel(X, Y : integer) : word) – возвращает значение, содержащее цвет пикселя с указанными координатами.

Line (X1, Y1, X2, Y2 : integer) – вычерчивает линию с указанными координатами начала и конца. Линия рисуется текущим стилем и цветом.

LineTo (X, Y : integer) – вычерчивает линию от текущего положения указателя до положения, заданного новыми координатами.

LineRel (Dx, Dy : integer) – вычерчивает линию от текущего положения указателя до положения, заданного приращениями его координат.

Bar (X1, Y1, X2, Y2 : integer) – построение прямоугольника по левой верхней и правой нижней точкам.

RectAngle(X1, Y1, X2, Y2 : integer) – построение прямоугольника без закраски.

DrawPoly (Num:word; Var Poly:PointType) – вычерчивает произвольную ломаную линию, заданную координатами точек излома.

FillPoly (Num:word; Var Poly:PointType) – рисует и закрашивает многоугольник в текущий цвет.

При вычерчивании дуг необходимо учитывать соотношение максимального значения x и y для конкретного терминала, чтобы окружность не выглядела эллипсом. Это можно осуществить с помощью процедуры:

GetAspectRatio(Var xa, ya : word) – возвращает два числа, позволяющие оценить отношение сторон графического экрана.

Масштабный коэффициент можно установить с помощью процедуры:

SetAspectRatio(xa, ya:word) – изменяет масштабный коэффициент отношения сторон графического экрана.

Circle (x,y:integer, R:word) – вычерчивание окружности цветом фона. **x,y** – координаты центра, **R** – радиус.

Arc (x,y:integer, StAngle, EndAngle, R:word) – вычерчивание дуги. **x,y** – координаты центра, **StAngle, EndAngle** – начальный и конечный углы, **R** – радиус.

FillEllipse (x,y:integer, xR, yR:word) – строится эллипс и закрашивается текущем стилем. **xR, yR** – горизонтальный и вертикальный радиусы. При **xR=yR** получается окружность.

Ellipse (x, y:integer, StAngle, EndAngle:word, xR, yR:word) – строится эллиптическая дуга.

PieSlice (x, y:integer, StAngle, EndAngle:word, R:word) – строится сектор по начальному и конечному углу и закрашивается текущим стилем заполнения.

Задержку изображения обеспечивают процедуры:

ReadKey – читать код нажатой клавиши.

Readln – пустой оператор чтения, ожидание нажатия клавиши ENTER.

Repeat – повторять, пока не выполнится какое-либо условие.

Delay – приостанавливает работу программы на указанное число миллисекунд.

18.5. Шаблон пользователя

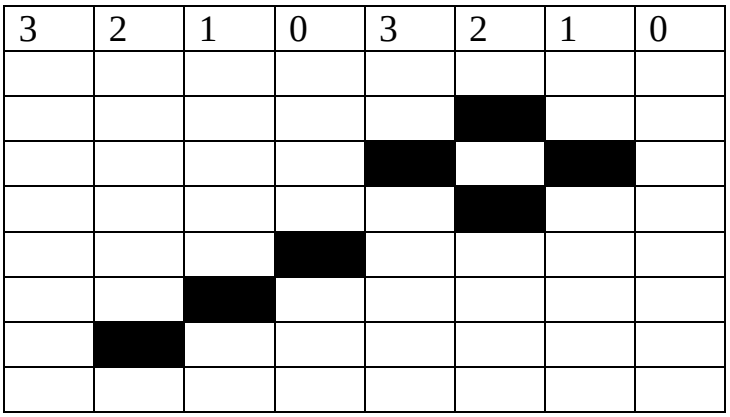
Шаблон – это матрица 8 на 8 битов. В 16-ричной системе используется по полбайта, куда в 2-й системе заносится 16-ричная конфигурация.

Схема соответствия:

10	2	16	10	2	16
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	**A**
3	0011	3	11	1011	**B**
4	0100	4	12	1100	**C**
5	0101	5	13	1101	**D**
6	0110	6	14	1110	**E**
7	0111	7	15	1111	**F**

Перед 16-ричной цифрой пишется $.

Пример:



Данный шаблон запишем по строкам сверху вниз ($00, $04, $0A, $04, $10, $20, $40, $00)

А может быть наоборот? Или я недопонимаю

SetLinePattern(Pattern:FillPatternType, Color : word) – устанавливает образец рисунка и цвет штриховки.

В программе используется следующим образом:

```
Program prim19;
Uses graph, crt;
Const a:fillpatterntype=($00, $04, $0a, $04, $10, $20, $40, $00);
Var gd,gm:integer;
Begin
    gd:=detect;
    initgraph(gd,gm,' ');
    setfillpattern(a,3);
    bar(0,0,100,100);delay(50000);
    closegraph;
End.
```

18.6. Установка спикера (динамика)

Одним из способов построения звуковых рядов является использование частот, соответствующих нотам.

Выдачу набора звуков можно организовать с помощью функции включения динамика, выключения динамика и задержки во времени.

SOUND(hnword) – включить динамик. **Hг** – частота в герцах.

NOSOUND – выключить динамик.

До	ре	ми	фа	соль	ля	си
262	294	330	349	392	440	494

Пример:

```
Program prim20;
Uses graph, crt;
Var gd,gm:integer;
```

```
Const  c=262; d=294; e=330; f=349; g=392; a=440; b=494;
Begin
Sound(c); delay( 10000);
Sound(d); delay( 10000);
Sound(e); delay( 10000);
Sound(f); delay( 10000);
Sound(g); delay( 10000);
Sound(f); delay( 10000);
Sound(e); delay( 10000);
Sound(d); delay(10000);
Sound(c); delay(10000);
Nosound;
End.
```

18.7. Черепашья графика

Может быть применена в игровых программах или при построении графиков физических и производственных процессов. Идея черепашьей графики связана с понятием «черепашки» на экране, которая, перемещаясь по экрану, рисует линии. Изменяя направление движения черепашки, можно рисовать кривые линии.

Функции работы с «черепашкой» возможны в модуле расширенной графики **Graph3.** «Черепашка» может быть видимой и невидимой. С «черепашкой» связано световое перо. Оно может быть поднято, тогда линия невидима, опущено, тогда линия видима.

ClearScreen – чистка экрана.

Home – перемещение «черепашки» в (0, 0).

SetHeading(Num) – поворот.

TurnLeft(Num) – поворот влево на **Num** градусов.

TurnRight(Num) – поворот вправо на **Num** градусов.

Back(Num) – перемещение назад на **Num** пикселей.

Forwd(Num) – перемещение вперед на **Num** пикселей.

PenDown – опустить перо.

PenUp – поднять перо.

HideTurtle – «черепашка» невидима.

ShowTurtle – «черепашка» видима.

SetPenColor(Num) – номер цвета для рисования линии.

18.8. Использование окон

Для организации работы с окнами нужен модуль **CRT**. Окно может быть графическое или текстовое. В случае работы с графическим окном нужен еще модуль GRAPH.

Window(xl, yl, x2, y2) – установка текстового окна, xl, yl – координаты верхнего левого угла, x2, y2 – координаты нижнего правого угла.

TextBackGround(Color) – установка цвета фона текстового окна.

GoToXY(x, y:word) – перемещение курсора внутри текстового окна, с отсчетом от начала окна.

SetViewPort(xl, yl, x2, y2, clip) – установка графического окна, xl, yl – координаты верхнего левого угла, x2, y2 – координаты нижнего правого угла. Параметр **clip** определяет, будет ли рисунок отсекаться при выходе за границу окна или нет. После создания окна за точку отсчета принимается верхний левый угол окна, имеющий координаты (0, 0).

ClearViewPort – очистка графического окна.

ClearDevice – очистка графического экрана.

19. ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа № 1
Знакомство с языком ПАСКАЛЬ

1. Запустите программную среду языка Паскаль, изучите главное меню, просмотрите все режимы.
2. Создайте новый файл (***File–New***).
3. Наберите текст программы:

```
program new;
uses crt;
{программу составил _____}
const pi=3.14;
var    r:integer; s:real;
begin
clrscr;
writeln(“Вычисление площади круга”);
write(“Введите радиус: “);
read(r);
s:=pi*sqr(r);
write(“Площадь равна”, s:6:2);
end.
```

4. Проверьте программу на ошибки (***F9***).
5. Сохраните файл с программой (***File–Save*** или ***F2***).
6. Запустите программу на выполнение (***Ctrl+F9***).
7. Посмотрите результат работы программы (***Alt+F5***).
8. Объясните каждую строчку программы. ***(25 баллов)***
9. Сохраните файл с программой под другим именем (***File–Save as***).
10. Измените текст программы для вычисления площади треугольника по формуле Герона.
11. Проверьте программу на ошибки (откомпилируйте).
12. Сохраните файл с программой.
13. Запустите программу на выполнение.

14. Посмотрите результат работы программы (***Alt+F5***).
15. Объясните каждую строчку программы. ***(35 баллов)***
16. Добавьте в программу условие для проверки, можно ли построить треугольник с заданными длинами сторон.
17. Проверьте программу на ошибки (откомпилируйте).
18. Сохраните файл с программой.
19. Запустите программу на выполнение.
20. Объясните каждую строчку программы. ***(40 баллов)***

Лабораторная работа №2
Основные операторы PASCAL: условный оператор

1. Запустите программную среду языка Паскаль.
2. Создайте новый файл (***File–New***).
3. Составьте программу:

1-вариант Даны три числа. Найти все пары взаимно-обратных чисел.

2-вариант Среди чисел *a, b* и *c* найти наименьшее и вывести его на терминал.

3-вариант Даны три числа. Найти все пары взаимно-противоположных чисел.

4-вариант Дана градусная мера угла. Установить, является ли угол острым, тупым или прямым.

5-вариант Определить, равна ли сумма двух первых цифр заданного четырехзначного числа сумме двух его последних цифр.

6-вариант Среди чисел *a, b* и *c* найти наибольшее и вывести его на терминал.

7-вариант Определить, является ли число N двузначным отрицательным числом.

8-вариант A, B, C – длины отрезков. Установить, могут ли они являться сторонами равностороннего треугольника.

9-вариант Используя признаки делимости, написать программу, которая проверяет, делится ли натуральное число k на 5 (последняя цифра 0 или 5).

10-вариант Определить, является ли натуральное число N нечетным трехзначным числом.

11-вариант Даны три числа. Если среди этих чисел есть положительное, то вывести его на терминал.

12-вариант Проверить, можно ли из четырех данных отрезков составить параллелограмм.

4. Проверьте программу на ошибки (***F9***).
5. Сохраните файл с программой (***File–Save*** или ***F2***).
6. Запустите программу на выполнение (***Ctrl+F9***).
7. Посмотрите результат работы программы (***Alt+F5***).
8. Объясните каждую строчку программы. ***(25 баллов)***
9. Создайте новый файл (***File–New***).

10. Составьте программу для вычисления функции:

1-вариант $y=\begin{cases}\ln(x+1), & x>1\\ \sin^2\sqrt{|ax|}, & x\le 1\end{cases}$ $x\in[0,5;2]$

2-вариант $y=\begin{cases}\sqrt{at^2+b\sin t+1}, & t<1,3\\ at+b, & t=1,3\\ \sqrt{at^2+b\cos t+1}, & t<1,3\end{cases}$ $t\in[0,8;2]$

3-вариант $y=\begin{cases}2ax+|x-1|, & x<0\\ \dfrac{e^x}{\sqrt{1+x^2}}, & x\ge 0\end{cases}$ $x\in[-1;3]$

4-вариант $y=\begin{cases}\sin x\cdot\ln x, & x>3,5\\ \cos^2 x, & x\le 3,5\end{cases}$ $x\in[2;5]$

5-вариант $y=\begin{cases}a\ln x+\sqrt{|x|}, & x>1\\ 2a\cos x+3x^2, & x\le 1\end{cases}$ $x\in[0;5]$

6-вариант $y=\begin{cases}x^2+\sqrt{x}, & x>3\\ \dfrac{4x-8}{x^4+1}, & x\le 3\end{cases}$

7-вариант $y=\dfrac{(x-3)^2\sqrt{x-2}\cdot x}{(x+1)^4(x+5)}$

8-вариант $y=\dfrac{3\sqrt{x+1}}{2-\sqrt{x}}$

9-вариант $y=\dfrac{2x^2}{\sqrt{x^3+1}}$

10-вариант $y=\dfrac{e^{-at}}{at+1}$

11-вариант $y=\dfrac{x^2+x}{\sin x}$

12-вариант $y=\dfrac{x^3-3ax}{9x-9}$

11. Проверьте программу на ошибки (***F9***).
12. Сохраните файл с программой (***File–Save*** или ***F2***).
13. Запустите программу на выполнение (***Ctrl+F9***).
14. Посмотрите результат работы программы (***Alt+F5***).
15. Объясните каждую строчку программы. ***(40 баллов)***
16. Создайте новый файл (***File–New***).

17. Составьте программу:

1-вариант Составить программу, которая выводит на терминал цвет радуги по его порядковому номеру.

2-вариант Составить программу, которая выводит на терминал название пальца руки по его порядковому номеру.

3-вариант Составить программу, которая выводит на терминал название дня недели по его порядковому номеру.

4-вариант Составить программу, которая выводит на терминал название месяца по его порядковому номеру.

5-вариант Дана градусная мера угла. Установить, является ли угол острым, тупым или прямым.

6-вариант Составить программу, которая по запросу номера дня недели выводит на терминал количество пар в этот день.

7-вариант Определить, является ли введенное число однозначным, двузначным, трехзначным, четырехзначным или пятизначным.

8-вариант Составить программу, которая анализирует человека по возрасту и относит к одной из четырех групп: дошкольник, ученик, работник, пенсионер.

9-вариант Составить программу, которая для целого числа k (1-99) напечатает фразу "Мне k лет", где k – введенное число. При этом в нужных случаях слово "лет" заменить словом "год" или "года".

10-вариант Составить программу, которая по его порядковому номеру дня недели определяет рабочий день или выходной.

11-вариант Составить программу, которая переводит десятизначное представление цифры в словесное.

12-вариант Составить программу, которая определяет время года по введенному номеру месяца.

18. Проверьте программу на ошибки (***F9***).
19. Сохраните файл с программой (***File–Save*** или ***F2***).
20. Запустите программу на выполнение (***Ctrl+F9***).
21. Посмотрите результат работы программы (***Alt+F5***).
22. Объясните каждую строчку программы. ***(35 баллов)***

Лабораторная работа №3
Основные операторы PASCAL: циклы

1. Запустите программную среду языка Паскаль.
2. Составьте программу вычисления среднего нескольких чисел:
    a. для заранее определенного количества чисел. ***(30 баллов)***
    b. для неизвестного количества чисел. ***(30 баллов)***
3. Проверьте программу на ошибки.
4. Сохраните файл с программой.
5. Запустите программу на выполнение.
6. Посмотрите результат работы программы.

7. Объясните каждую строчку программы.
8. Создайте новый файл.
9. Составьте программу для вывода на терминал таблицы значений функции ***(40 баллов)***:

1-вариант $y = \ln x \quad x \in (0;5] \quad \Delta x = 0,5$

2-вариант $y = e^{2x} - 4x \quad x \in [1;5] \quad \Delta x = 0,5$

3-вариант $y = \dfrac{x-3}{x^2+1} \quad x \in (0;3] \quad \Delta x = 0,5$

4-вариант $y = \dfrac{x^2+1}{x^4+5} \quad x \in [-7;-5] \quad \Delta x = 0,5$

5-вариант $y = \dfrac{1,2-9,8x}{x^2+3} \quad x \in [10;12] \quad \Delta x = 0,2$

6-вариант $y = \ln(x-5] \quad x \in [8;10] \quad \Delta x = 0,4$

7-вариант $y = x^2 - 3x^{10} \quad x \in [-2;1] \quad \Delta x = 0,5$

8-вариант $y = tg\ x \quad x \in \left(0; \dfrac{\pi}{2}\right) \quad \Delta x = 0,1$

9-вариант $y = ax^2 + 1 \quad x \in [-3;-1] \quad \Delta x = 0,2$

10-вариант $y = \sin(x+3] \quad x \in [10;13] \quad \Delta x = 0,2$

11-вариант $y = 4x - 2\cos x \quad x \in [0;1] \quad \Delta x = 0,1$

12-вариант $y = x^2 - 8\sqrt[3]{x} \quad x \in [3;6] \quad \Delta x = 0,5$

10. Проверьте программу на ошибки.
11. Сохраните файл с программой.
12. Запустите программу на выполнение.
13. Посмотрите результат работы программы.
14. Объясните каждую строчку программы.

Лабораторная работа №4
Массивы

1. Запустите программную среду языка Паскаль.
2. Составьте программу, где организован ввод чисел в массив с терминала и вычисляется разница между максимальным и минимальным элементами массива. На терминал вывести максимальный элемент, минимальный элемент и разницу между ними. ***(25 баллов)***
3. Проверьте программу на ошибки.
4. Сохраните файл с программой.
5. Запустите программу на выполнение.
6. Посмотрите результат работы программы.
7. Объясните каждую строчку программы.
8. Создайте новый файл

9. Составьте программу ***(30 баллов)***:

1-вариант В матрице заменить элементы главной диагонали их квадратами. Вывести на экран измененную матрицу.

2-вариант Найти максимальный элемент матрицы и заменить им элементы последней строки.

3-вариант Вычислить сумму элементов первой строки и заменить этой суммой элементы главной диагонали. Вывести измененную матрицу на экран.

4-вариант В матрице заменить элементы последней строки нулями. Вывести измененную матрицу на экран.

5-вариант Заменить элементы последнего столбца на число 10. Вывести измененную матрицу на экран.

6-вариант В матрице все отрицательные элементы главной диагонали заменить их модулями. Вывести измененную матрицу на экран.

7-вариант Найти сумму положительных элементов матрицы и заменить ею элементы первой строки. Вывести измененную матрицу на экран.

8-вариант Дана матрица размерностью N×M. Переставляя строки и столбцы, добиться, чтобы минимальный элемент находился в верхнем левом углу.

9-вариант Увеличить элементы главной диагонали на 3 и вывести измененную матрицу на экран.

10-вариант Заменить элементы предпоследнего столбца значением (–1). Вывести измененную матрицу на экран.

11-вариант Дана матрица размерностью N×M. Элементы, индексы которых четные, возвести в квадрат. Измененную матрицу вывести на экран.

12-вариант Найти минимальный элемент матрицы и заменить им элементы главной диагонали. Вывести измененную матрицу на экран.

10. Проверьте программу на ошибки.
11. Сохраните файл с программой.
12. Запустите программу на выполнение.
13. Посмотрите результат работы программы.
14. Объясните каждую строчку программы.
15. Создайте новый файл.
16. Составьте программу для подсчета слов в предложении ***(15 баллов).*** Выведите на терминал самое длинное слово ***(30 баллов)***.
17. Проверьте программу на ошибки.
18. Сохраните файл с программой.
19. Запустите программу на выполнение.
20. Посмотрите результат работы программы.
21. Объясните каждую строчку программы.

Лабораторная работа №5
Сортировка массива

1. Запустите программную среду языка Паскаль.
2. Составьте программу, в которой организована *линейная сортировка массива* по следующему алгоритму ***(25 баллов)***:
   2.1. Ищется наименьший элемент массива и помещается на место первого;
   2.2. Рассматривается укороченный массив без первого элемента, с ним выполняется действие, описанное в 1.1, и т.д.
3. Проверьте программу на ошибки.
4. Сохраните файл с программой.
5. Запустите программу на выполнение.
6. Посмотрите результат работы программы.
7. Объясните каждую строчку программы.
8. Создайте новый файл.
9. Составьте программу, в которой организована сортировка массива методом *«Пузырек»* ***(25 баллов)***:
10. Проверьте программу на ошибки.
11. Сохраните файл с программой.
12. Запустите программу на выполнение.
13. Посмотрите результат работы программы.
14. Объясните каждую строчку программы.
15. Создайте новый файл.
16. Изучите другие методы сортировки массива. Опишите алгоритм любого метода и составьте программу для реализации этого метода. ***(50 баллов)***
17. Проверьте программу на ошибки.
18. Сохраните файл с программой.
19. Запустите программу на выполнение.
20. Посмотрите результат работы программы.
21. Объясните каждую строчку программы.

Лабораторная работа №6
Работа с массивами от записи

1. Запустите программную среду языка Паскаль.
2. Составьте программу, в которй создается массив от записи со структурой:

1-вариант

Фамилия	Адрес	Номер телефона
A(15)	X(12)	X(8)

2-вариант

Наименование товара	Цена	Количество
A(20)	9(8),9(2)	9(8)

3-вариант

Название поезда	Номер	Время прибытия	Стоянка
A(20)	9(3)	X(5)	9(2)

4-вариант

Фамилия автора	Название книги	Год издания
A(15)	A(20)	9(4)

5-вариант

Название газеты	Количество подписчиков	Цена газеты
A(20)	9(4)	9(6), 9(2)

6-вариант

Фамилия абонента	Год установки	Номер телефона
A(16)	9(4)	X(8)

7-вариант

Марка автомобиля	Номер	Цена	Ф.И.О. владельца
X(14)	X(6)	9(10), 9(2)	X(22)

8-вариант

Шифр материала	Наименование материала	Наличие материала на складе
9(4)	X(12)	9(4)

9-вариант

Табельный номер	Фамилия	Номер цеха	Должность
9(4)	A(17)	9(3)	A(15)

10-вариант

Фамилия	Лицевой счёт	Номер цеха	Заработная плата
A(15)	9(8)	9(3)	9(6),9(2)

11-вариант

Ф.И.О. студента	Группа	Оценка
A(18)	X(6)	9(1)

12-вариант

Номер рейса	Номер билета	Вес багажа
X(6)	9(3)	9(6), 9(3)

3. Заполните массив, организуйте вывод всех данных и поиск по критерию. При организации поиска предусмотрите вывод сообщения, если данные отсутствуют. ***(60 баллов)***
4. Организуйте меню ***(40 баллов)***, состоящее из пунктов:
   - Ввод данных
   - Вывод всех данных
   - Поиск по критерию
   - Выход
5. Проверьте программу на ошибки.
6. Сохраните файл с программой.
7. Запустите программу на выполнение.
8. Посмотрите результат работы программы.
9. Объясните каждую строчку программы.

Лабораторная работа №7
Файлы прямого доступа

1. Запустите программную среду языка Паскаль.
2. Составьте программу создания файла прямого доступа со структурой, описанной в лабораторной работе №6.
3. Организуйте просмотр на терминале всего файла, а также просмотр записи по запросу ее номера. ***(60 баллов)***
4. Организуйте меню ***(40 баллов)***, состоящее из пунктов:
    1. Ввод данных
    2. Вывод всего файла
    3. Просмотр записи по номеру
    4. Выход
    5. Проверьте программу на ошибки.
    6. Сохраните файл с программой.
    7. Запустите программу на выполнение.
    8. Посмотрите результат работы программы.
    9. Объясните каждую строчку программы.

Лабораторная работа №8
Текстовые файлы

1. Запустите программную среду языка Паскаль.
2. Составьте программу создания чтения файла, созданного редактором, со структурой, описанной в лабораторной работе №6.
3. Организуйте просмотр на терминале всего файла. Вывод данных оформить в виде таблицы с заголовком. ***(60 баллов)***
4. Организуйте меню ***(40 баллов)***, состоящее из пунктов:
    1. Ввод данных
    2. Вывод всего файла
    3. Просмотр записи по номеру
    4. Выход
5. Проверьте программу на ошибки.
6. Сохраните файл с программой.
7. Запустите программу на выполнение.
8. Посмотрите результат работы программы.
9. Объясните каждую строчку программы.

Лабораторная работа №9
Процедуры и функции

Постановка задачи:

Имеется выборка значений некоторых элементов в количестве **N (N<100) – $X_1, X_2,...,X_N$.**

Найти среднее значение:

$$M=(X_1+X_2+...+X_N)/N$$

Определить дисперсию (вариацию):

1. Для размещения элементов используйте рабочий массив:

LIST:ARRAY [1..100] OF REAL;

Ввод элементов массива с терминала.

2. Составить подпрограммы:

READLIST(NUM, VAR LIST) – процедура ввода элементов с терминала, где **NUM** (количество элементов массива) – входной параметр процедуры, передается из головной программы; **LIST** (заполненный массив) – выходной параметр, передается из процедуры в головную программу.

MED(LIST) – функция для подсчета среднего значения, где **LIST** (массив) – входной параметр, передается из головной программы; функция возвращает среднее значение.

3. В головной программе организуйте обращение к подпрограммам и вычисление дисперсии.

Головная программа выдает сообщения:
"СРЕДНЕЕ ЗНАЧЕНИЕ:" XXXX.XX
"ДИСПЕРСИЯ:" XXXX.XX

4. Запустите программную среду языка Паскаль.
5. Наберите текст программы.
6. Проверьте программу на ошибки.
7. Сохраните файл с программой.
8. Запустите программу на выполнение.
9. Посмотрите результат работы программы.
10. Объясните каждую строчку программы. ***(100 баллов)***

Лабораторная работа №10
Использование модуля UNIT

1. Запустите программную среду языка Паскаль.
2. Составьте программу, в которой: ***(100 баллов)***
   1) вводятся элементы матрицы;
   2) вызывается процедура преобразования элементов;
   3) выводятся элементы полученной матрицы.
3. Модуль UNIT:
   Организовать процедуру, в которой отрицательные элементы матрицы заменяются нулями. Входными параметрами процедуры являются элементы матрицы, в головную программу передается матрица в измененном виде.
4. Наберите текст программы.
5. Проверьте программу на ошибки.
6. Сохраните файл с программой.
7. Запустите программу на выполнение.
8. Посмотрите результат работы программы.

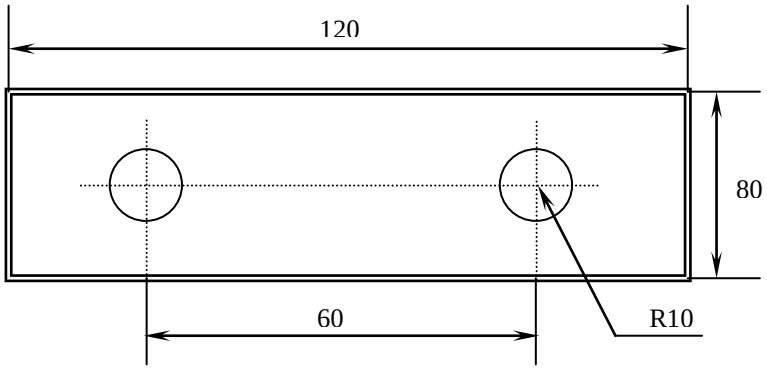
9. Объясните каждую строчку программы.

Лабораторная работа №11
Указатели и списки
Организация стека с помощью списковой структуры

1. Запустите программную среду языка Паскаль.
2. Напишите программу организации калькулятора, работающего со стековой структурой по принципу обратной польской последовательности.
   - Стек организован в виде списковой структуры, загрузку в стек и выгрузку элементов оформить процедурами.
   - Элементы-числа в контрольном примере разделены пробелами, элементы-знаки – знаками табуляции.
   - Ввод данных из файла на диске, вывод результата на терминал.
3. Войдите в систему и наберите текст программы.
4. Откомпилируйте программу и запустите ее на выполнение.
5. Объясните каждую строчку программы.

Лабораторная работа №12
Графика

1. Составьте программу построения чертежа детали «ПЛАНКА». Цвет закраски должен отличаться от цвета фона ***(50 баллов)***.



2. Войдите в систему и наберите ткст программы.
3. Откомпилируйте программу и запустите ее на выполнение.
4. Объясните каждую строчку программы.
5. Составьте программу построения окружности с расширением радиуса. Начальный радиус 10. Координаты центра совпадают с центром экрана. Цвет фона, линии и внутренней закраски различны и меняются при изменении радиуса ***(50 баллов)***.
6. Войдите в систему и наберите текст программы.
7. Откомпилируйте программу и запустите ее на выполнение.
8. Объясните каждую строчку программы.

20. ВОПРОСЫ ДЛЯ ПОДГОТОВКИ К ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

1. Понятие структурного программирования.
2. Принципы структурного программирования.
3. Виды структур, используемые в структурном программировании.
4. Определение идентификатора.
5. Алфавит языка Паскаль.
6. Структура Паскаль–программы.
7. Ввод и вывод данных.
8. Классификация типов данных (простые, стандартные, порядковые).
9. Целый тип данных, операции, определенные над этим типом.
10. Вещественный тип данных, операции, определенные над этим типом.
11. Литерный (символьный) тип данных, операции, определенные над этим типом.
12. Логический тип данных, операции, определенные над этим типом.
13. Скалярный тип данных, операции, определенные над этим типом.
14. Оператор присваивания.
15. Условные операторы.
16. Операторы цикла.
17. Оператор безусловного перехода.
18. Понятие вектора.
19. Логическая структура вектора.
20. Физическая структура вектора.
21. Дескриптор.
22. Ввод и вывод элементов массива (одномерного, двумерного).
23. Обращение к элементу массива (одномерного, двумерного).
24. Описание строки.
25. Обращение к элементам строки.
26. Процедуры и функциии для работы со строками.
27. Понятие массива (матрицы).
28. Логическая структура двумерного массива.
29. Физическая структура двумерного массива.
30. Сортировка массива.
31. Виды сортировки (внутренняя, внешняя).
32. Определение записи.
33. Структура записи.
34. Обращение к элементам записи.
35. Оператор присоединения имен.
36. Структура записи с вариантами.
37. Способы описания массивов от записей.
38. Обращение к элементам массива от записи.
39. Как в программе можно определить количество компонент файла?
40. Какой смысл файловой и буферной переменной?

41. Методы организации обработки файлов.
42. Текстовые файлы (определение, описание).
43. Процедуры и функции для работы с текстовыми файлами.
44. Двоичные файлы (определение, описание).
45. Процедуры и функции для работы с двоичными файлами.
46. Файлы без типа (определение, описание).
47. Процедуры и функции для работы с файлами без типа.
48. Понятие процедуры и функции. Сходства и отличия.
49. Локальные и глобальные переменные.
50. Понятие рекурсии.
51. Формальные и фактические параметры.
52. Способы передачи параметров.
53. Внешние процедуры (определение, описание).
54. Модуль UNIT (разделы).
55. Польская запись алгебраических формул.
56. Статические и динамические переменные. Свойства.
57. Виды списковых структур.
58. Указатели.
59. Процедуры и функции для работы с указателями и адресами.
60. Назначение графического режима.
61. Инсталляция графического режима.
62. Относительная и абсолютная системы координат.
63. Установка курсора.
64. Использование окон.
65. Установка цвета, стиля, шаблона.
66. Функции рисования точек, линий, фигур.
67. Шаблон пользователя.

21. ЗАДАЧИ ДЛЯ ПРАКТИЧЕСКИХ И САМОСТОЯТЕЛЬНЫХ ЗАНЯТИЙ

21.1. Выражения

1. Записать на Паскале следующие формулы:
   - $\sqrt{1+x^2}$
   - $|a+bx|$
   - $\log_2 x/5$
   - $\cos^2 x^3$
   - x^{-1}
   - x^{100}
   - $\sqrt[3]{1+x}$
   - $\sqrt[8]{x^8+8^x}$

2. Вычислить значение выражения:
   - ***trunk(6.9)***
   - ***round(6.9)***
   - ***trunk(6.2)***
   - ***round(6.2)***
   - ***trunk(-1.8)***
   - ***round(-1.8)***
   - ***round(0.5)***
   - ***round(-0.5)***
   - ***8 div 6 mod 4 * 2***
   - ***8 div (6 mod 4) * 2***
   - ***8 div 6 mod (4 * 2)***
   - ***8 mod (6 mod 4 * 2)***
   - ***18 mod 7 * 2 – 3 mod 7***
   - ***6 mod 7 + 5 div 4 * 2***
   - ***6 mod (7 + 5) div 4 * 2***
   - ***6 mod ((7 + 5) div 4) * 2***
   - ***x:=15 div (8 mod 3);***
     y:=17 mod x * 5 – 19 mod 5 * 2;
   - ***x:= 2*5 div 3 mod 2;***
     y:= 2*5 div (3 mod 2);
     x;= x * y;
     y:= y * y;
   - ***3 * 7 div 2 mod 7 / 3 – trunk(sin(1))***
   - ***20 div 6***
   - ***20 mod 6***
   - ***20 div 4***
   - ***20 mod 4***
   - ***2 div 5***
   - ***2 mod 5***
   - ***12 div 0***
   - ***3.0 mod 3***
3. Чему равно занчение выражения в Паскале – ***(1/3*3–1)***?
4. Записать на Паскале выражение – $\boldsymbol{Sin(x^o)}$.
5. Присвоить целой переменной ***H*** вторую, третью, четвертую от конца цифру в записи натурального числа ***K***. Например, для записи третьей цифры, если *K=63815*, то *H=8*.
6. С какими значениями ***x*** следующие равенства будут выполняться?
   - ***x div 5 = x mod 5***
   - ***20 div x = 20 mod x***
   - ***x div 5 = 8***
   - ***50 div x = 7***
7. Указать порядок выполнения операций:
   - ***a and b or not c and d***
   - ***a + b and not c or b and c + b * a***
   - ***x >= 0 or t and x or y * x***
8. Записать на Паскале выражение, истинное при выполнении указанного условия и ложное в противном случае:
   - $x \in (2; 5)$
   - $x \notin (2; 5)$
   - $x \in [-1; 1]$

- $x \notin [-1; 1]$
- $x \in (-1; 1)$
- ***хотя бы одно из трех чисел положительно***
- ***ни одно из трех чисел не является положительным***

9. Возвести выражение в степень, используя наименьшее количество операций сложения и вычитания (только + и –):
   - x^7
   - $(x + y)^8$
   - x^9
   - $(a + 1)^{29}$
   - $(a + b + c)^{32}$
   - $(a + b)^{100}$
10. Вычислить a^6, a^8 за три операции умножения, a^9, a^{10} за четыре операции умножения.
11. Используя для вывода каждой цифры один оператор вывода, написать блок программы, отображающий на экране следующие цифры:
    1 2 3 4
    5 6
    7 8
    9
12. Написать программу, которая вычисляет периметр и площать правильного 17-угольника, вписанного в окружность заданного радиуса.
13. Определить, равна ли сумма первых двух цифр заданного четырехзначного числа сумме двух его последних цифр. Не использовать условный оператор ***IF***.
14. Определить, есть ли среди цифр заданного трехзначного числа одинаковые. Не использовать условный оператор ***IF***.
15. Даны три произвольных числа. Записать выражение, которое позволит определить, можно ли построить треугольник с такими длинами сторон. Не использовать условный оператор ***IF***.

21.2. Условные операторы

1. Вычислить значение выражения: $y = \begin{cases} \sin x, & x > 0 \\ \cos x, & x \le 0 \end{cases}$
2. Вычислить значение выражения: $y = \begin{cases} 4x^2, & x < 1 \\ \dfrac{x^2}{\sqrt[3]{x}}, & 1 \le x < 5 \\ \ln x, & x \ge 5 \end{cases}$

3. Вычислить значение выражения: $y = \begin{cases} 4AB - 8x^2, & x < A \\ \dfrac{2x^2 - 5}{x^4 + 2} + B, & A \le x < B \\ \dfrac{A - B}{2x^2 + 3}, & x \ge B \end{cases}$
4. Составить программу для вычисления функции, учитывая область ее определения: $y = \dfrac{x+3}{2-x}$.
5. Определить, является ли натуральное число *N* четным двузначным числом.
6. Составить программу для вычисления корней квадратного уравнения: $ax^2+bx+c=0$.
7. Установить, являются ли три введенных целых числа пифагоровой тройкой, т.е. $a^2+b^2=c^2$.
8. ***A, B, C*** – длины отрезков. Установить, могут ли они являться сторонами равнобедренного треугольника.
9. Используя признаки делимости, написать программу, которая проверяет, делится ли натуральное число k на 4 (две последние цифры нули или образуют число, делящееся на 4).
10. Каждое из чисел *a* и *b* отлично от нуля. Если числа с одинаковым знаком, то заменить меньшее из них большим. Если числа имеют разные знаки, то присвоить каждому из них знак числа меньшего по абсолютной величине.
11. Вычислить стоимость покупки с учетом скидки. Если стоимость покупки до 1000 рублей – скидка 5%, до 10000 рублей – 10%, свыше 10000 рублей – 15%.
12. Составить программу, определяющую, пройдет ли график функции $y=5x^2-7x+2$ через точку с координатами ***(a, b)***.
13. К финалу конкурса были допущены трое претендентов. Соревнования проходили в три тура. Первый конкурсант набрал в первом туре ***m1*** баллов, во втором – ***n1*** баллов, в третьем – ***p1*** баллов. Второй конкурсант – соответственно ***m2***, ***n2***, ***p2***. Третий – ***m3***, ***n3***, ***p3***. Определить, сколько баллов набрал победитель.
14. Составить программу, определяющую, входит ли введенная с клавиатуры цифра в запись введенного натурального трехзначного числа.
15. Даны два числа. Проверить, имеют ли они одинаковый знак.
16. Определить, равен ли квадрат заданного трехзначного числа кубу суммы цифр этого числа.
17. Определить, есть ли среди первых трех цифр из дробной части заданного положительного вещественного числа цифра 0.
18. Определить, есть ли среди цифр заданного трехзначного числа одинаковые.

19. Составить программу, которая выводит агрегатное состояние воды (пар, жидкость, лед), в зависимости от введенной температуры.
20. Написать программу калькулятор. С терминала вводятся два числа и знак арифметической операции, выводится результат вычислений.

21.3. Циклы

21.3.1. Цикл For

1. Найти сумму и количество всех четных положительных чисел, вводимых с терминала.
2. Составить программу для нахождения всех натуральных делителей натурального числа N.
3. Найти все целые числа, кратные 3, на интервале от k до n.
4. Дано n чисел. Найти количество тех, которые делятся на 5 и не делятся на на 7.
5. Найти квадраты всех отрицательных чисел среди N заданных.
6. Дано целое N>0. Вывести на экран таблицу значений переменных x, y, z, таких, что $n=x^2+y^2+z^2$.
7. Вычислить n!
8. Найти количество чисел, среди N введенных, которые имеют нечетные порядковые номера, а сами являются четными числами.
9. Составить программу для вычисления суммы:
   - ***1*1! + 2*2! + 3*3! + … +n*n!***
   - ***1*3 + 3*5 + 5*7 + …+ n(n+2)***
   - ***1 + 11 + 111 + … +111…1-n слагаемых***
10. Дана последовательность из N вещественных чисел. Определить, со скольких отрицательных значений она начинается.
11. Дано N чисел. Найти два числа, которые меньше всех остальных.
12. Дана последовательность из N чисел. Определить, содержит ли последовательность хотя бы два равных соседних элемента.
13. Одноклеточная амеба каждые 3 часа делится на 2 клетки. Определить, сколько клеток будет через 3, 6, …, 24 часа. Час вводится с терминала.
14. Найти все двузначные числа, которые делятся на сумму своих цифр.
15. Найти все трехзначные натуральные числа, средняя цифра которых равна сумме первой и третьей цифр.
16. Вычислить сумму ряда:
    - $S = \cos(1) + 2 \cdot \cos(2) + 3 \cdot \cos(3) + \ldots + n \cdot \cos(n) = \sum_{i=1}^{n} i \cdot \cos(i)$
    - $S = \sqrt{2} - 1 + \sqrt{3} - 1 + \ldots + \sqrt{n} - 1 = \sum_{i=1}^{n} (\sqrt{i} - 1)$
17. Дана последовательность чисел. Найти среднее арифметическое всех четных отрицательных чисел.
18. Дана последовательность чисел. Найти произведение всех четных отрицательных чисел.

19. Дана последовательность чисел. Найти произведение всех нечетных положительных чисел.
20. Дана последовательность чисел. Найти среднее арифметическое всех нечетных отрицательных чисел.

21.3.2. Циклы While, Repeat

1. Вычислить значения функции $Y = 4 \cdot X^3 - 2 \cdot X^2 + 5$ для значений x, изменяющихся от -3 до 1, с шагом 0.1.
2. Вычислить значения функции $y = 2 \cdot \sqrt[4]{|x|}$ для значений x, изменяющихся от -7 до -2, с шагом 0.3.
3. Составить таблицу перевода центнеров в килограммы для промежутка от 1ц до 2 ц с шагом 0.1.
4. Составить таблицу перевода дюймов в сантиметры, для промежутка от 5 до 10 с шагом 0.5 (1 дюйм=2.5 см).
5. Дана последовательность чисел $x_1, x_2, x_3, ..$, где каждое $x_i = i / \sqrt{i+1}$. Найти произведение тех чисел X, номера которых кратны 3 (т.е. $P = x_3 \cdot x_6 \cdot x_9 \cdot ....$).
6. Дана последовательность чисел $x_1, x_2, x_3, ..$, где каждое $x_i = i / \sqrt{i+1}$. Найти сумму чисел X с нечетными номерами (т.е. $S=x_1+x_3+x_5+...$).
7. Вычислить значение функции $y = \cos(x^3) \cdot \sin(x)$ для значений x, изменяющихся от n до m, с шагом 0.1.
8. Составить таблицу перевода футов в сантиметры для промежутка от 1 до 5 с шагом 0,5 (1 фут = 30,4 см).
9. Написать программу, которая вычисляет сумму и количество цифр, составляющих данное натуральное число.
10. Составить программу, которая будет вычислять сумму чисел, введенных с терминала, до тех пор, пока сумма не станет больше 30.
11. Вывести на экран число, получаемое выписыванием в обратном порядке цифр заданного натурального числа.
12. Используя признаки делимости, написать программу, которая проверяет, делится ли число на 3, т. е. сумма цифр кратна 3.
13. Автоморфным называется число, цифры которого совпадают с последними цифрами его квадрата (6^2=36, 25^2=625, 76^2=5776). Составить программу для нахождения автоморфных чисел в интервале [n;m].

21.4. Массивы

21.4.1. Одномерные массивы

1. В одномерном массиве целых чисел найти сумму элементов, больших 5.
2. В одномерном массиве вещественных чисел определить количество элементов, равных 7.
3. В одномерном массиве целых чисел найти произведение положительных однозначных элементов массива.

4. В одномерном массиве вещественных чисел найти среднее арифметическое отрицательных элементов массива.
5. В одномерном массиве, состоящем из k вещественных чисел, все отрицательные элементы увеличить на 5. Измененный массив вывести на экран.
6. В одномерном массиве, состоящем из m вещественных чисел, элементы больше первого уменьшить в два раза. Измененный массив вывести на экран.
7. В одномерном массиве целых чисел все элементы, меньше последнего, увеличить в два раза. Измененный массив вывести на экран.
8. В одномерном массиве, состоящем из n целых чисел, каждый второй элемент обнулить. Измененный массив вывести на экран.
9. В одномерном массиве, состоящем из m вещественных чисел, заменить первый элемент на минимальный. Измененный массив вывести на экран.
10. В одномерном массиве, состоящем из m вещественных чисел, заменить последний элемент на максимальный. Измененный массив вывести на экран.
11. В одномерном массиве определить, сколько одинаковых пар соседних элементов.
12. В одномерном массиве найти сумму и количество тех элементов, порядковые номера которых кратны 3, а сами элементы меньше 0.
13. В одномерном массиве найти сумму чисел, расположенных между максимальным и минимальным элементами.
14. В одномерном массиве найти сумму элементов с четными порядковыми номерами.
15. Создать массив из N элементов, в котором первые два элемента вводятся с терминала, а все последующие равны сумме двух предыдущих.
16. Если среди элементов массива нет нулей, то обнулить каждый третий элемент, иначе возвести в квадрат все отрицательные элементы.
17. Вывести на экран все двузначные элементы массива.
18. Вывести на экран те элементы массива, индексы которых являются степенями двойки (1, 2, 4, 8, 16, …).
19. В одномерном массиве поменять местами числа, так чтобы получить массив в обратном порядке.
20. Даны два упорядоченных массива. Объединить элементы этих массивов в один так, чтобы он остался упорядоченным.
21. Даны два массива. Найти наименьший элемент первого массива, который не входит во второй массив (считая, что хотя бы один такой элемент есть).
22. Дан символьный массив. Вывести на экран сначала все цифры, входящие в массив, затем все буквы.
23. Дан массив символов. Вывести на экран только строчные буквы русского алфавита.

24. Дан символьный массив. Определить, сколько различных литер (символов) входит в этот массив.
25. Дан массив строчных латинских букв. Вывести на экран в алфавитном порядке все буквы, которые входят в этот массив по одному разу.

21.4.2. Двумерные массивы

1. Дана матрица размерностью N×N. Вывести на экран элементы побочной диагонали.
2. Дана матрица размерностью N×M. Вычислить среднее арифметическое элементов первой строки матрицы.
3. Во втором столбце изменить знак элемента на противоположный. Вывести измененную матрицу на экран.
4. Дана матрица размерностью N×M. Вычислить сумму элементов каждого столбца.
5. Найти сумму минимальных элементов главной и побочной диагоналей.
6. Определить, является ли матрица симметричной относительно главной диагонали.
7. Дана матрица размерностью N×M. Сформировать два одномерных массива. В один занести четные элементы, в другой – нечетные. Полученные массивы вывести на экран.
8. Поменять местами максимальный и минимальный элементы массива.
9. Составить программу сложения двух матриц.
10. Удалить строку двумерного массива, в которой впервые встретился элемент, равный p. Новый массив вывести на экран.
11. Поэлементно вычесть последний столбец из всех столбцов, кроме последнего.

21.4.3. Строки

1. Дана строка символов. Подсчитать количество символов ‘A’ в строке.
2. Дана строка символов. Определить, каких букв больше, ‘o’ или ‘a’.
3. Дана строка, состоящая из произвольных символов. Проверить, входят ли цифры в строку.
4. Дано слово. Проверить, входит ли первая буква ещё раз в это слово.
5. Дано слово. Проверить, симметрично ли оно.
6. Дана строка символов. После каждого символа ‘к’ вставить ‘и’.
7. Дана строка. Удалить из нее все заранее определенные сочетания символов (например, все сочетания "на").
8. Дана строка символов. Удалить из строки все гласные буквы.
9. Дана строка символов. Проверить, есть ли в ней повторяющиеся буквы.
10. Дана строка символов. Вывести на экран слово, в котором каждая буква встречается дважды.
11. Дана строка символов. Вывести на экран первые буквы всех слов.
12. Дана строка символов. Все слова вывести на экран в столбик.

13. Дана строка символов. Выделить подстроку между первой и второй точками.

21.5. Массив от записи

1. Массив от записи имеет структуру:

   Фамилия – A(15)

   Сумма взноса – 9(4).9(2)

   Организовать ввод **N** элементов и найти общую сумму взносов.
2. Массив от записи содержит ведомость успеваемости студентов за три по трем предметам.
   - Организовать ввод данных об N студентах.
   - Вывести фамилии студентов, имеющих задолженность хотя бы по одному предмету. В случае отсутствия таких студентов записей вывести соответствующее сообщение.
   - Вывести название предмета, который был сдан лучше всего других.
3. Массив от записи содержит сведения об участии студентов в спортивных соревнованиях.
   - Организовать ввод данных об N студентах.
   - По запросу фамилии вывести сведения о студенте. В случае отсутствия такой фамилии вывести соответствующее сообщение.
4. Дан массив от записи, содержащий сведения о студентах. Отсортировать данные по алфавиту.

21.6. Файлы

1. Файл имеет структуру:

   Код товара

   Наименование

   Цена

   Номер склада

   - Создать файл.
   - По запросу наименования товара вывести сведения о нем.
   - По запросу номера склада вычислить общую стоимость товаров.
   - Вывести на экран десятую запись, если такая есть, иначе вывести соответствующее сообщение.
2. На диске имеется список товаров и их цены. Создать файл прямого доступа и выдать на терминал строку под номером 15.
3. Последовательность вещественных чисел находится на диске в виде текстового файла. Написать программу ввода чисел и определения количества чисел, больших числа A.
4. Составить программу вычисления N значений переменной X по рекуррентной формуле $X_k=3X_{k-1}+5$ при $X_0=0$. Результат расчета вывести в двоичный файл.

5. Найти среднее значение ряда вещественных чисел, находящихся на диске в виде двоичного файла.
6. Найти максимально число среди N чисел, записанных в двоичный файл.
7. Написать программу, которая считает количество пустых строк в текстовом файле.
8. Написать программу, которая выводит на экран строку текстового файла с максимальной длиной.
9. Дан текстовый файл, состоящий из строк. Создать другой файл, состоящий из строк первого, дополненных точками до самой длинной строки первого файла.
10. Создать файл целых чисел. Отсортировать числа в файле.
11. Создать файл вещественных чисел. Написать программу, переписывающую компоненты файла в обратном порядке. Новый файл не создавать.

21.7. Процедуры и функции

1. Написать функцию, которая вычисляет объем цилиндра. Входными параметрами функции являются радиус и высота цилиндра. В головной программе организовать ввод радиуса и высоты, вызов функции и вывод объема.
2. Написать функцию, которая удаляет начальные пробелы из строки, полученной в качестве аргумента.
3. По вещественному ***x*** вычислить значение выражения: $\boldsymbol{\log_5 x * tg(x+1) - tg^2(2+\log(x+1))}$.
4. Даны длины отрезков ***a, b, c, d***. Для каждой тройки этих отрезков, из которых можно построить треугольник, вычислить площадь треугольника. Процедура выводит значение площади или сообщение о том, что треугольник построить нельзя.
5. Даны координаты вершин двух треугольников. Определить, какой из них имеет большую площадь.
6. Описать рекурсивную функцию, которая подсчитывет количество цифр в числе.
7. Даны три матрицы. Вывести на терминал ту из них, которая содержит больше нулевых элементов. Подсчет нулевых элементов оформить подпрограммой.
8. Даны две матрицы. Написать подпрограмму, вычисляющую сумму элементов главной диагонали. Вывести на экран матрицу с наименьшей суммой.
9. Описать процедуру, формирующую текстовый файл, состоящий из 9 строк:

    1
    22
    333

…

Файл передается в качестве входного параметра.

10. Дан текстовый файл. Написать функцию для подсчета числа строк, которые начинаются с буквы ***d***.
11. Текстовый файл содержит последовательность вещественных чисел. Описать функцию нахождения наибольшего из них. Файл передается в качестве входного параметра, максимальное число – выходной параметр. Ввод и вывод организовать в головной программе.
12. Во входном файле задана непустая последовательность положительных вещественных чисел, за которой следует отрицательное число. Описать рекурсивную функцию без параметров для нахождения суммы этих положительных чисел.

21.8. Списковые структуры

1. Данные о студентах (фамилия, оценка) вводятся с терминала. На основе этих данных создать списковую структуру и вывести ее на экране.
2. Дан текстовый файл. Используя списковую структуру, отобразить на экране содержимое файла в обратном порядке.
3. Написать программу, которая заменяет в списке все вхождения элемента x на элемент y.
4. Найти максимальный элемент в списке вещественных чисел.
5. Написать программу, которая по списку целых чисел строит два новых списка:
   - положительных элементов;
   - отрицательных элементов

   Вывести три списка на терминал.

21.9. Графика

1. Написать программу, которая вычерчивает на экране ломаную линию, состоящую из 200 звеньев, окрашенных в различные цвета. Цвет и координаты звеньев выбираются случайным образом.

2. Написать программу, которая выводит на экран контур пятиконечной звезды.

3. Нарисовать изображение шахматной доски.

4. Вывести на экран гистограмму успеваемости учеников класса по итогам контрольной работы.

СПИСОК ИСПОЛЬЗОВАННОЙ И РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Алексеев Е.Р., Чеснокова О.В., Кучер Т.В. Free Pascal и Lazarus: Учебник по программированию. ДМК Пресс, 2010. 438 с. http://e.lanbook.com/books/element.php?pl1_id=1267
2. ГОСТ 19.002-80. Схемы алгоритмов и программ. Правила выполнения.
3. ГОСТ 19.003-80. Схемы алгоритмов и программ. Обозначения условные графические.
4. ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
5. Давыдова Н.А. Боровская Е.В. Программирование: учебное пособие. «Бином. Лаборатория знаний», 2012. 238 с. http://e.lanbook.com/books/element.php?pl1_cid=25&pl1_id=8764
6. Златопольский Д.М. Программирование: типовые задачи, алгоритмы, методы. «Лаборатория знаний», (ранее "БИНОМ. Лаборатория знаний") 2012. 223 с. http://e.lanbook.com/books/element.php?pl1_id=8765
7. Медведик В.И. Практика программирования на языке Паскаль (задачи и решения). Учебное пособие. ДМК Пресс, 2013. 590 с. http://e.lanbook.com/books/element.php?pl1_id=58700
8. Обухович Т.М., Сахань Ю. В. Программирование на языке высокого уровня. ПАСКАЛЬ: Учебное пособие для студентов специальности 230100 «Информатика и вычислительная техника» / Рубцовский индустриальный институт.- Рубцовск, 2010.- 78 с.
9. Окулов С.М. Алгоритмы обработки строк. «Лаборатория знаний» (ранее "БИНОМ. Лаборатория знаний"), 2015. 258 с. http://e.lanbook.com/books/element.php?pl1_id=66113
10. Окулов С.М. Основы программирования. "Лаборатория знаний" (ранее "БИНОМ. Лаборатория знаний"), 2015. 339 с. http://e.lanbook.com/books/element.php?pl1_id=66119
11. Окулов С.М. Основы программирования "Бином. Лаборатория знаний", 2012. 336 с. http://e.lanbook.com/books/element.php?pl1_cid=25&pl1_id=8783
12. Тишин В.И. Программирование на Паскале: практикум. "Бином. Лаборатория знаний", 2013. 364 с. http://e.lanbook.com/books/element.php?pl1_cid=25&pl1_id=8774

Татьяна Михайловна Обухович
Людмила Анатольевна Попова

Программирование. Паскаль

Учебное пособие для студентов направления
«Информатика и вычислительная техника»

Редактор Е.Ф. Изотова

Подписано к печати 25.12.15. Формат 60x84 1/16.
Усл. печ. л. 4,63. Тираж 20 экз. Зак. 151518. Рег. №148.

Отпечатано в ИТО Рубцовского индустриального института
658207, Рубцовск, ул. Тракторная, 2/6.